Lecture 16, May 4, 2026

ECS 235B, Foundations of Computer and Information Security

Spring Quarter 2026

Security Considerations

- Security analysis must rest on specification of current system, not previous ones or changes only
  - If modification specifications are only ones, security analysis based upon incomplete specifications
  - If previous system has full security specifications, then analysis may be complete

Security Specifications

- Used when design specifications adequate except for security issues
- Develop supplemental specifications to describe missing security functionality
  - Develop document that starts with security functions summary specification
  - Expand to address security issues of components, subcomponents, modules, functions

Example: System X

- Underlying UNIX system completely specified, including complete functional specifications and internal design specifications
  - Neither covered security well, let alone document new functionality
- Team supplemented existing documentation with security architecture document
  - Addresses deficiencies of existing documentation
  - Gives complete overview of each security function
  - Additional documentation describes external interface, internal design of all functions

Formal Specifications

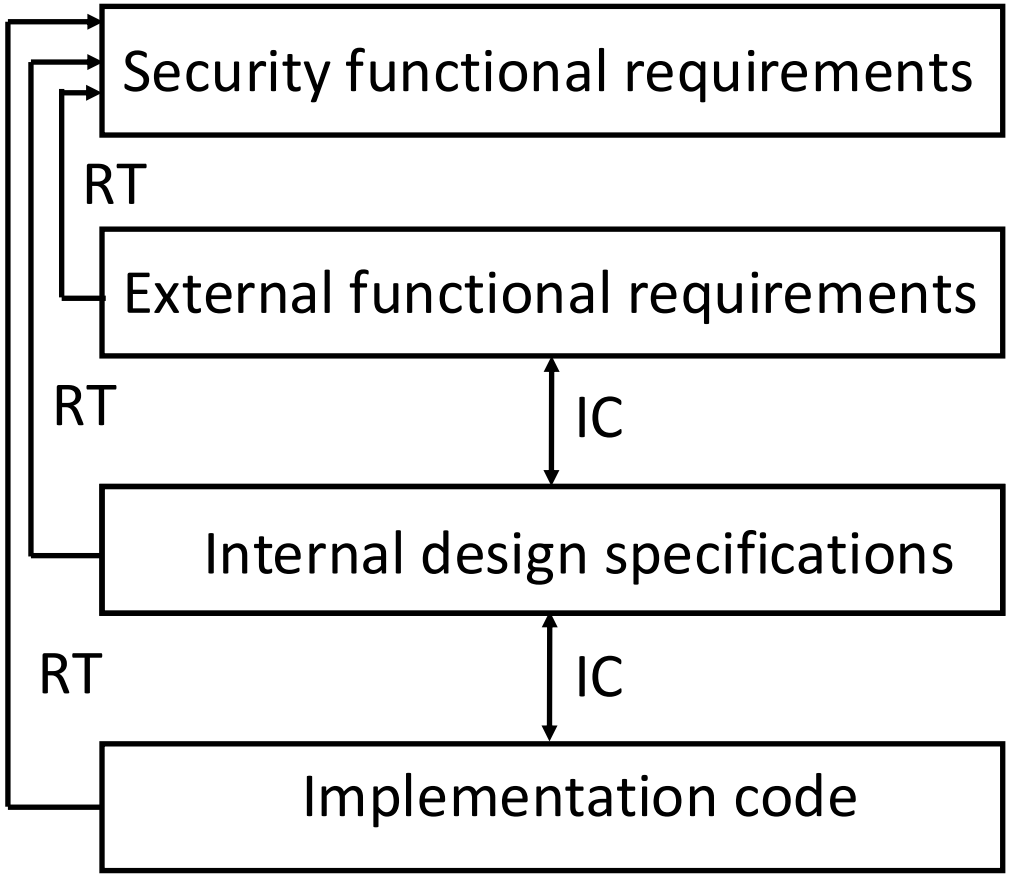
- Any specification can be formal
- Written in formal language, with well-defined syntax and sound semantics
- Supporting tools allow checking
  - Parsers
  - Theorem provers

Justifications

- Formal techniques
  - Proofs of correctness, consistency
- Informal techniques
  - *Requirements tracing*: showing which specific security requirements are met by parts of a specification
  - *Informal correspondence* (also called *representation correspondence*): showing a specification is consistent with adjacent level of specification

Requirements Mapping and Informal Correspondence

RT: requirements tracing
IC: Informal correspondence



Mappings Between Layers

- Informal techniques most appropriate when all levels of specification have identified requirements and all adjacent pairs of specifications have been shown to be consistent
  - Security functions summary specification and functional specification
  - Functional specification and high-level design specification
  - High-level design specification and low-level design specification
  - Low-level design specification and implementation code
- Doing third mapping may be difficult as difference in levels of abstraction can obscure relationship
  - Intermediate level often simplifies this

Example

- Family of specifications across several levels
- Security requirement R2 requires users of system be identified to system, and to have identification authenticated by system before use of any system functions
- Identification and authentication (I&A) high-level security-enforcing function from security functions summary specification:
    1. Users identify themselves to system using *login_ID* before they can use any system resources
    2. Users use password to authenticate their identity; system must accept password as authentic before any resources can be used
    3. Password must meet specific size, character constraints
- Interfaces *login*, *change_password* described in functional specification

Example

- Requirements mapping represented by table following explanation
  - In this example, only R2 maps to I&A
- Informal correspondence between functional, security functions summary specifications are:
  - *login* maps to items 1, 2 in description of I&A
  - *change_password* maps to items 2, 3 in description of I&A

Security requirements	**Function 1**	**I&A**	**…**	**Function *m***
R1				
R2		X		
…				
R*n*				

Informal Arguments

- Requirements tracing identifies components, modules, functions that meet requirements but not how well they are met
- *Informal arguments* uses approach similar to mathematical proofs

Example

- System W is a new version of an existing product
  - Previous version had good requirements, security functions summary, external functional, and design specifications
- System W added bug fixes, features (some large and pervasive)
- Developers created external functional specification, internal design specification documents for all modifications of the system
  - Each document defined scope to be modifications only
- Security analysts asked developers many questions
- Resulting combined security specification and analysis document addressed impacts of change on security of previous system

Example (*con't*)

- Analysis document contained
  - Security analysis document containing individual documents for each of the different functional areas
  - System overview document
  - Test coverage analysis document

Example (*con't*)

- Documentation semiformal, written in natural language with code excepts where practical
  - Design overview: gave high-level description of component, relevant security issues, impact on security
  - Requirements section: identified security functionality in module, traced it to applicable security functional requirements
  - Interface analysis: described new or impacted interfaces, mapped requirements to them, identified and documented security problems and made recommendations

Formal Methods

- Requirements tracing checks specifications satisfy requirements
- Specifiers intend to process specification using automated tools
  - Proof-based technology typically based on some form of logic (like predicate calculus); user constructs proof, proof checkers validate it
  - Model checking takes a security model and processes a specification to determine if it meets the model's constraints

Reviews of Assurance Evidence

- Reviewers given guidelines for review
- Other roles:
  - Scribe: takes notes
  - Moderator: controls review process
  - Reviewer: examines assurance evidence
  - Author: author of assurance evidence
  - Observer: observe process silently
- Important: managers may *only* be reviewers, and only then if their technical expertise warrants it

Setting Review Up

- Moderator manages review process
  - If not ready, moderator and author's manager discuss how to make it ready with author
  - May split it up into several reviews
  - Chooses team, defines ground rules
- Technical Review
  - Reviewers follow rules, commenting on any issues they uncover
    - May request moderator to stop review, send back to author
  - General and specific comments to author

Review Meeting

- Moderator is master of ceremonies
  - Grammatical issues presented first
  - General and specific comments next
  - Goal is to collect comments on entity, *not* to resolve differences
  - Scribes write down comments and who made it (anyone can see it, help scribe, verify comment made)

Conflict Resolution

- After meeting, scribe creates Master Comment List
  - Reviewers mark “Agree” or “Challenge”
  - All comments that everyone “Agree”s with are put on Official Comment List (OCL)
  - Rest must be resolved by reviewers
- Moderator, reviewers then:
  - Accept as is
  - Accept with changes on OCL
  - Reject

Conflict Resolution

- Author takes OCL, makes changes as sees fit
- Author then meets with reviewers
  - Explains how each comment made by reviewer was handled
  - All must be resolved to satisfaction of author, reviewer
- Review completed

Informal Review

- Occurs sometimes due to quick pace of releases, bug fixes
  - Review process does not include moderator or scribe
  - Review may use electronic communications with one reviewer

Implementation Considerations for Assurance

- Make system modular, with minimum of interfaces
  - Interfaces are well-designed
  - Remove any non-security functionality from them, whenever possible
- Choice of programming language can affect assurance
  - Use one providing built-in features to avoid common flaws
    - Strong typing, built-in checks for buffer overflow, data hiding, error handling, etc.
  - Otherwise, develop and use appropriate coding standards and guidelines
    - Useful, but limited support for good code

Implementation Management

- *Configuration management*: control of changes made in system's components, documentation, and testing throughout development, operational life
- Need processes, tools to do this effectively
- Configuration management system consists of:
  - Version control and tracking
  - Change authorization: restrict change check in to authorized people
  - Integration procedures
  - Product generation tools: generate the distribution version from authorized version

Justification

- Goal is to demonstrate implementation meets design
- Security testing
- Formal methods: used during coding processes, work best on small parts of a program performing well-defined tasks
  - We'll discuss these later (next chapter)

Testing

- Testing techniques
  - *Functional* (*black box*): testing to see how well entity meets its specifications
  - *Structural* (*white box*): testing based on analysis of code to develop test cases
- When to do testing
  - *Unit testing*: testing by developer on code module before integration
    - Usually structural testing
  - *System testing*: functional testing performed by integration team on integrated modules
    - May include structural testing
  - *Third-party* (*independent*) testing: functional testing by a group outside development organization

Security Testing

- Testing that addresses product security
  - *Security functional testing*: functional testing specific to security issues described in relevant specification
    - Focus is on pathological cases, boundary value issues, and so forth
  - *Security structural testing*: structural testing specific to security implementation found in relevant code
  - *Security requirements testing*: security functional testing specific to security requirements found in requirements specification
    - May overlap significantly with security functional testing

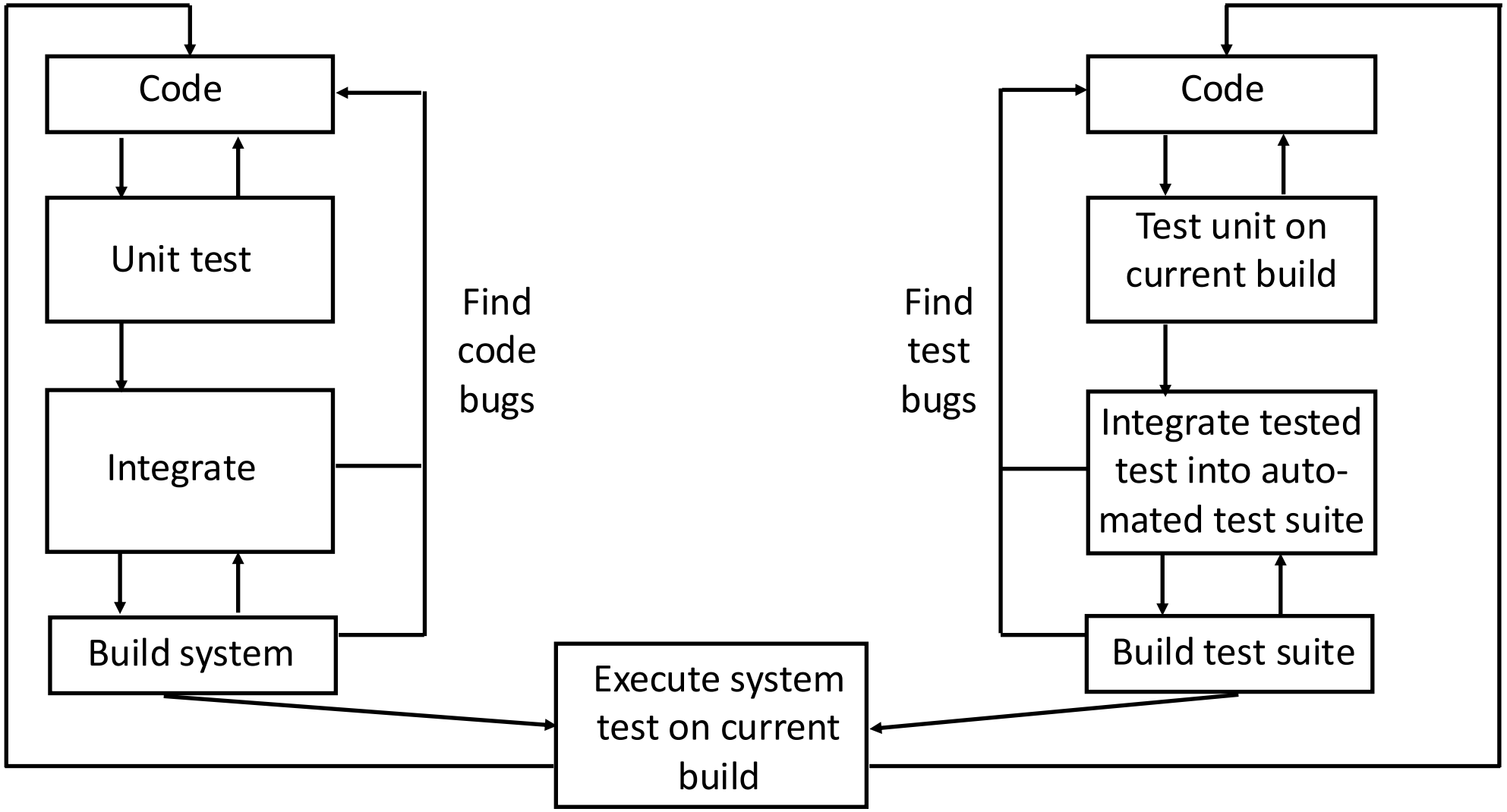
Security Testing

- Test coverage covers system security functions more consistently than ordinary testing
  - When completed, provides rigorous argument that all external interfaces have been completely tested

Security Testing

- Usually takes place at external interface level
  - Here, “interface” is point at which processing crosses security perimeter
  - Users access system through these
  - Therefore, violations of policy occur through these
- Parallel efforts, one by programming team, other by test team
- Security test suites very large
  - Automated test suites essential

Code Development and Testing



Plans and Reports

- Configuration management, documentation very important
  - Testers develop, document test plans, test specifications, test procedures, test results
- Writing test plans, specifications, procedures help authors examine, correct approaches
  - Provides assurance about test methodology
  - Enables analysis of test suite for correctness, completeness
- Reports identify which tests entity has passed, which it has failed
  - Watch out for failures due to automation (where automated test fails, but same test run independently of suite passes)

Security Testing Using PGWG

- PAT (Process Action Team) Guidance Working Group developed systematic approach to test development using successive decomposition of system, requirements tracing
- Methodology works well in system defined into successively smaller components
  - Requirements mapped to successively lower levels of design using *test matrices*
  - At lowest level, *test assertions* claim interfaces meet each requirement
  - Used to develop test cases
  - Includes documentation approach

PGWG Test Matrices

- High level matrix
  - Rows are entity subsystems, major components
  - Columns are high-level security areas focused on functional requirements
    - Like access controls, integrity controls, cryptography
  - Cells give pointers to relevant documentation, lower-level test matrices
- Low level matrix
  - Rows are interfaces to subsystem, component
  - Columns represent security areas, their subdivisions, individual requirements
  - Cells contain test assertions, each of which apply to single interface and requirement
    - Any empty cells must be justified to show why requirement does not apply

Example: Testing Security-Enhanced UNIX

- System includes file, memory, process, and IPC management, process control, I/O interfaces and devices
- Security functional requirement areas
  - Discretionary access control
  - Privileges, identification, authentication (I&A)
  - Object reuse protection
  - Security audit
  - System architecture constraints
- Testing uses interpretation of PGWG methodology
  - High-level matrix
  - Low-level matrices, 1 for each row of high-level matrix

Example: High-Level Matrix

Security Requirement Area						
Component	DAC	Priv	I&A	OR	Audit	Arch
Process management					✓	
Process control	✓	✓		✓	✓	✓
File management	✓	✓		✓	✓	✓
Audit subsystem		✓	✓	✓	✓	✓
I/O subsystem interfaces	✓	✓	✓	✓	✓	
I/O device drivers		✓		✓	✓	✓
IPC management	✓	✓		✓	✓	✓
Memory management	✓	✓		✓	✓	✓

Example: Low-Level Matrix

System Call	DAC u/g/o	DAC ACL	Priv	I&A	OR	Security Audit	Logging	Isolation	Protection Domains
brk					✓			✓	✓
madvise								✓	✓
mmap	✓	✓			✓	✓	✓	✓	✓
mprotect	✓	✓				✓		✓	✓
msync								✓	✓
munmap			✓		✓	✓		✓	✓
plock	✓	✓	✓		✓	✓		✓	✓
vm-ctl	✓	✓	✓		✓	✓	✓	✓	✓

Test Assertions

- Created by identifying security-relevant, testable, analyzable conditions
  - Review design documentation for this
- PGWG methods for stating assertions
  - Develop statements describing behavior that must be verified
    - Example: “Verify that the calling process needs DAC write access permission to the parent directory of the file being created. Verify that if access is denied, the return error code is 2.”
  - Develop statements that tester must prove or disprove with tests
    - Example: “The calling process needs DAC write access permission to the parent directory of the file being created, and if access is denied, it returns error code 2.”
  - State assertions as claims embedded within structured specification format

Test Specifications

- Test cases to verify truth of each assertion for each interface
- PGWG suggests:
  - High-level test specifications (HLTS) describe, specify test cases for each interface
  - Low-level test specifications (LLTS) provide information about each test case
    - Like setup and cleanup conditions, other environmental conditions

Example: HLTS for Interface *stime*()

High-level test specification includes assertion, test case specifications

Assertion Number	Requirement Area and Number	Assertion	Relevant Test Cases
1	PRIV AC_1	Verify that only root can use system call *stime*() successfully	Stime_1, 2
2	PRIV AC_2	Verify audit record generated for every failed *stime*() call	Stime_1, 2
3	PRIV AC_3	Verify audit record generated for every successful *stime*() call	Stime_1, 2

Test Case Specifications

- Describe specific tests required to meet assertions

Test Case Name and Number	Is UserID = *root*?	Expected Results
Stime_1	Yes	Call to *stime*() should succeed; audit record should be generated noting successful attempt and new clock time
Stime_2	No	Call to *stime*() should fail; audit record should be generated noting failed attempt

LLTS for Stime_1

Test case name: K_MIS_stime_1

Test case description: Call *stime* as a non-*root* user to change system time; this should fail, verifying only *root* can use this call successfully

Expected result: *stime* call should fail with return value of –1, system clock should be unchanged, error number (*errno*) set to **EPERM**, audit record as shown below

Test specific setup:

1. Log in as a non-*root* user (*secusr1*)
2. Get the current system time

LLTS for Stime_1 (*con't*)

Algorithm:

1. Do the setup as above
2. Call *stime* to change system time to 10 min ahead of current time
3. If return value is –1, error number is **EPERM**, and current system time not new time given to *stime*, declare the test passed; otherwise, declare failed

Cleanup: If system time has changed, reduce current time to 10 minutes

LLTS for Stime_1 (*con't*)

Audit record field values for failure (success):

Authid	secusr1
RUID	secusr1
EUID	secusr1
RGID	scgrp1
EGID	secgrp1
Class	tune
Reason	Privilege failure (success)
Event	SETTHETIME_1
Message	Privilege failure (none)

Operation, Maintenance Assurance

- Bugs will be found during operation, requiring fixes
  - *Hot fix*: handle bugs immediately, sent out as quickly as possible
    - Used to fix bugs that immediately affect system security or operation
  - *Regular fix*: handle less serious bugs or give long-term solutions to bugs fixed by hot fix, usually collected until some condition arises and then sent out
    - Sent out as maintenance release or as “patch Tuesday” or some other way
- Well-defined procedures handle, track reported flaws
  - Include information about bug, such as description, remedial actions, severity, pointer to related configuration management entries, other documentation
  - Actions taken follow same security procedures used during original devlopment

Evaluation Outline

- Goals of formal evaluation
- Trusted Computer Security Evaluation Criteria (TCSEC), 1983–1999
- International Efforts and the ITSEC, 1991–2001
- Commercial International Security Requirements, 1990, 1991
- Federal Criteria, 1992
- FIPS 140, 1994–*present*
- Common Criteria, 1998–*present*
- SSE-CMM, 1997–*present*

Goals of Formal Evaluation

- Provide evidence that a system meets specific security requirements under specific conditions
- Evaluation methodology consists of:
  - Set of requirements defining security functionality
  - Set of assurance requirements giving steps for showing the system meets its functional requirements; these usually specify required evidence of assurance
  - Methodology for determining the system meets functionakl requirements based on analysis of assurance evidence
  - Measure of results of evaluation indicating how trustworthy the system is with respect to security requirements
    - Called *level of trust*

Deciding to Evaluate

- Certification needed due to government acquisition regulations
- Cost-benefit analysis
  - Requestors typically pay evaluator's charge and staffing costs
  - Interaction with evaluator can affect development, delivery schedules
- Certification for non-government use
  - Demonstrate trustworthiness of product

Historical Note

- Governments, militaries drove creation of security evaluation processes
  - They wanted to use commercial products rather than develop their own
  - So evaluation methodologies were applied to commercial systems also
- Evaluation methodologies have different structures
  - Some list requirements, use them to build trust categories
  - Others list requirements only in the description of trust categories

Trusted Computer System Evaluation Criteria (TCSEC)

- Also called the *Orange Book*
  - Expanded to include networks, databases, etc. in the *Rainbow Series*
- Provides set of criteria for evaluating security of products
- Emphasis is on confidentiality; integrity addressed indirectly through *-property
  - Emphasized confidentiality of government classified data
  - Availability not considered

TCSEC Evaluation

- 6 different evaluation classes: A1, B3, B2, B1, C2, C1
  - D class is for products that attempted evaluation but didn't fall into any of the other classes
- *Rated product*: a product that has been evaluated
- TCSEC organized by evaluation class
  - Defines functional, assurance requirements for each

TCSEC Functional Requirements

- *Discretionary access controls*: access control mechanism allowing controlled sharing of named objects by named entities
  - Propagation of access rights, ACLs, granularity of controls
- *Mandatory access controls*: embody Bell-LaPadula model
  - Label hierarchy, subject labels reflect authorizations and come from approvals (eg, security clearances), object labels reflect protection requirements
  - Required at B1 or higher
- *Label*: enable enforcement of mandatory access controls
  - Exporting of labeled information, labeling human-readable output, accurately represent clearances and classifications
  - Required at B1 or higher

TCSEC Functional Requirements

- *Identification and authorization*: users must identify themselves to the system and the system must authenticate that identity before the user can use the system
  - Granularity of authentication data, protection of that data, associating identity with auditable actions
- *Object reuse*: revocation of access rights when object released, and ensuring a new user cannot read previous contents of object when it is reused
- *Trusted path*: communications channel guaranteed to be only between user, TCB
  - Required at B2 or higher

TCSEC Functional Requirements

- *Audit*: existence of audit mechanism, protection of audit data
  - What audit records must contain, what events are to be recorded
- System architecture: include reference validation mechanism, process isolation, well-defined user interfaces, least privilege
- Operational assurance: trusted facility management including separation of operator, administrator roles at B2 or higher; trusted recovery procedures at A1; hardware diagnostics to validate hardware, firmware elements of TCB

TCSEC Assurance Requirements

- *Configuration management*: identification of configuration items, consistent mappings among documentation and code tools for generating TCB
  - Required at B2 or higher
- *Trusted distribution*: integrity of mapping between masters, distributed media; acceptance procedures
  - Required at A1
- *System architecture*: mandate modularity, minimization of complexity and other techniques for keeping TCB as small, simple as possible
  - Required at C1; increase until B3, where TCB must be full RVM

TCSEC Assurance Requirements

- *Design specification and verification*: these vary greatly among evaluation classes
  - C1, C2: none
  - B1: informal security policy model shown consistent with its axioms
  - B2: formal security policy model proven consistent with its axions, and system has descriptive top level specification (DTLS)
  - B3: DTLS shown consistent with security policy model
  - A1: system has formal top level specification (FTLS), approved formal methods show FTLS consistent with security policy model, also mapoping between FTLS and source code

TCSEC Assurance Requirements

- *Testing*: address conformance with claims, resistance to penetration, correction of flaws, search for covert channels
  - Requires use of formal methods in search at higher evaluation classes
- *Product documentation*: Security Features User's Guide includes description of protection mechanisms, how they interact, how to use them; Trusted Facility Manual is for administrators and says how to run product securely
  - Required at all evaluation classes; increases as level of classes increase

The Evaluation Classes

- C1, Discretionary Protection
  - Minimal functional requirements for identification, authentication, and discretionary access controls
  - Minimal assurance requirements for testing, documentation
  - Used only briefly, at beginning of the use of TCSEC
- C2, Controlled Access Protection
  - Functional requirements include object reuse, auditing
  - Assurance requirements require more stringent security testing
  - Most commonly used class for commercial products

The Evaluation Classes

- B1, Labeled Security Protection
  - Functional requirements include mandatory access controls, possibly restricted to specified set of objects, labelling to support this
  - Assurance requirements include more stringent security testing, informal model of security policy shown to be consistent with its axioms
- B2, Structured Protection
  - Functional requirements include mandatory access controls for all objects, labeling expanded, trusted path for login, enforcement of least privilege
  - Assurance requirements include covert channel analysis, configuration management, more stringent documentation, formal security policy model proven to be consistent with its axioms

The Evaluation Classes

- B3, Security Domains
  - Functional requirements include implementation of dull RVM, additional requirements for trusted path, constraints on code development (modularity, simplicity, layering, data hiding, etc.)
  - Assurance requirements include all of B2 requirements, more stringent testing, more requirements on DLTS, administrator's guide, design documentation
- A1, Verified Protection
  - Functional requirements are same as for B3
  - Assurance requirements include using formal methods in covert channel analysis, design specification, verification, correspondence between code and FTLS, as well as trusted distribution and increased test and design document requirements

The Evaluation Process

- Evaluators were government sponsored
- Evaluations had no fees for vendors
- Three phases
  - Application
  - Preliminary technical review
  - Evaluation

The Evaluation Process

- Application phase: vendor applied for evaluation
  - If government did not need product, application could be denied
- Preliminary technical review phase: discussions of evaluation process, schedules, development process, etc.
  - This determined when to provide evaluation team, and the basic evaluation schedule
- Evaluation phase: 3 parts, each part's results presented to technical review board (TRB), which approved that part before next part began
  - Design analysis part
  - Test analysis part
  - Final review

The Evaluation Process

- Design analysis part: rigorous review of system design based on provided documentation
  - Source code not reviewed
  - Stringent requirements on completeness, correctness of documentation
  - Initial product assessment report produced in this part
- Test analysis part: test coverage assessment, vendor supplied tests run
- Final review part: after approval of previous parts, a final evaluation report produced and given to TRB
  - When that approved final evaluation report, rating awarded

Ratings Maintenance Program (RAMP)

- Maintained assurance for new versions of evaluated product
- Vendor updated assurance evidence
- TRB reviewed report; when approved, new version given evaluation rating
- Vendor had to have trained Vendor Security Analyst on staff to perform RAMP process
- Not all enhancements were accepted by RAMP
  - For example, structural changes could require new evaluation

Impacts

- TCSEC was first evaluation technology
  - Created new approach to determining how secure a product is
  - Developed ideas of evaluation classes, assurance requirements, assurance-based evaluation
  - Technical depth of evaluation came from strength of foundation of requirements and classes, rigor of evaluation process, rigor of review
- Issues with TCSEC
  - Evaluation process difficult, often lacked enough resources
  - Functionality, assurance blended together in evaluation classes
  - Limited scope

Limitations of Scope

- Written for operating systems and does not translate well to other types of systems
- Focused on needs of US government
- Did not address integrity, availability, other business-critical applications
- National Computer Security Center developed criteria for other systems based on TCSEC
  - Trusted Network Interpretation (TNI), released in 1987
  - Trusted Database Management System Interpretation (TDI), released in 1992
  - Not many evaluations under these

Limitations of Process

- Requirements defining evaluation classes gradually expanded
  - Called *criteria creep*
  - Sometimes had to interpret requirements to apply them to specific products; these were published as informal addenda
  - Class requirements became union of TCSEC requirements and applicable interpretations
  - So as time passed, systems had to meet more stringent requirements

Limitations of Process

- Evaluations took too long
  - Many vendors misunderstood depth of evaluation, required interactions with evaluation teams
  - Way evaluations were done caused misunderstandings, scheduling problems
  - Vendors often lacked motivation to complete free evaluation
- Some evaluations took so long, product's end of life came before evaluation completed
- Towards end of TCSEC, government approved commercial labs as evaluators; charged a fee for evaluation
  - Reduced time problem with evaluations completing in around a year

Contributions

- Provided a process for security evaluation of products
  - Helped commercial sector realized need for computer security
- Its inadequacies led to development of new approaches and methodologies for evaluation

Information Technology Security Evaluation Criteria (ITSEC)

- Many countries created their own evaluation criteria
  - Canada, France, Germany, the Netherlands, the United Kingdom
  - Not reciprocal, so one product evaluated by each separately
- European Union standard developed to harmonize all these criteria
  - Result: ITSEC, published in 1991, EU endorsed it in 1995
  - Used until mid-2000s, until Common Criteria developed
- Different approach than TCSEC

Evaluation Basics

- Vendor provided functional criteria
  - *Security target* (ST) defined security functional criteria
  - Advantage: ITSEC could be used on any type of system
- *Target of evaluation* (TOE) is system, associated documentation, that is subject of evaluation
- UK defined exemplary sets of functional requirements
  - Systems certified as functional class and assurance class (eg, FC2-E3)

Assurance Requirements

- Defined within constraints of evaluation levels
- Effectiveness requirements for security target included:
  - Suitability of requirements: addressed consistency and coverage of security target
  - Binding of requirements: analyzed security requirements, mechanisms that implemented them

Assurance Requirements

- Requirements for TOE
  - Assessment of security measures used for *developer* environment during development, maintenance of TOE
  - Correspondence must be defined between all levels of representation in TOE
  - Required source code at several levels, object code at highest level
  - Distribution requirements at all levels
  - Vulnerability analysis required at design level
  - Ease of use analysis examined how system might be misused based on study of system documentation
  - Strength of mechanisms effectiveness requirement applied to each mechanism whose strength could be measured

Evaluation Levels

- E1, E2, E3, E4, E5, E6; E0 for products not meeting other levels

E1: requires ST, informal description of system, testing of system to show it satisfied ST

E2: E1 + informal description of detailed design, configuration control, distribution control process, evidence of testing

E3: E2 + more stringent requirements on detail design, correspondence between source code and security requirements

Evaluation Levels

E4: E3 + formal model of security policy, structured approach to design, design level vulnerability analysis

E5: E4 + correspondence between detailed design and source code, source code level vulnerability analysis

E6: E5 + use of formal methods

- Example: architectural design must be stated formally, shown to be consistent with formal model of security policy

Evaluation Process

- Each country had its own methodology for doing evaluations
  - This is the UK methodology
- Certified licensed evaluation facilities (CLEFs) evaluated for a fee
  - In turn, these certified by UK government
  - Also did consulting to help vendors prepare for evaluation
- Began with evaluation of security target (ST); once ST approved, product evaluated against the ST
- Certificate maintenance scheme required plan, evidence to support correct implementation of plan

Process Limitations

- Some considered using same company for evaluation preparation and the evaluation itself a conflict of interest
  - Different divisions of company, but could still have similar biases
- Usually 1 or 2 people made the decisions, and review of them was insufficient
- No body of experts to approve evaluator design analysis and test coverage analysis
  - Government body provided final approval of evaluation, but generally followed recommendation of evaluation team

Vendor-Provided Security Targets

- Vendors often did not have the expertise to develop appropriate security targets
  - Usually work of 1 or 2 people
  - No official review assessed quality of the ST
  - These were ameliorated by use of predefined functionality classes
- So some concern that ITSEC evaluations did not check that claims made sense
  - Just verified product met claim

Impacts

- Evaluation allowed flexibility in defining requirements and mixing functional and assurance requirements
- Use of commercial labs made evaluation process quicker
- Methodology allowed any type of product to be evaluated
- ITSEC evaluations often considered weaker than those of TCSEC
  - Development of functional requirements has potential weaknesses
  - Evaluation process itself not so rigorous as that of TCSEC
- No reciprocity of evaluations with US, Canadas

Federal Criteria

- NSA, NIST developed Federal Criteria (FC) to replace TCSEC with new evaluation approach
  - FC had catalogue of functional requirements
- *Protection profiles* (PP) identified requirements, other information particular to family of systems
  - An abstract specification of security aspects of an IT product
  - Product independent, describing range of products
  - Functional, assurance requirements bound together with rationale describing threats, intended method of use

FC Requirements

- Catalogue of functional requirements
  - All functional requirements of TCSEC
  - Requirements from CSIR included system entry constraints, others
  - Requirements for resource allocation, fault tolerance (availability)
  - Requirements for security management
- Assurance requirements
  - Met both TCSEC, ITSEC requirements
  - New assurance requirement for life cycle process

Impacts

- Contributed concept of protection profile
  - PP requirements selected from FC functional requirements catalogue
- PP included
  - Information for identification, cross-referencing
  - Description of problem that profile addressed
  - Rationale portion included threats, environment, assumptions, justification
- FC supported evaluation of protection profiles
- Development of profile registry making FC-approved PPs for general use

FIPS 140: Cryptographic Modules

- Standard for evaluating cryptographic modules
  - Sponsored by NIST, Canadian Security Establishment under the Cryptographic Module Validation Program (CMVP)
- ”Module” is set of hardware, firmware, software that implements cryptographic logic or processes
  - If done in software, processor included in cryptographic module
  - Evaluation of software modules includes operating system
- Cryptographic Algorithm Validation Program (CAVP) provides for evaluation of approved crypto algorithms against specific algorithm specifications
  - List of approved crypto algorithms is dynamic
  - CMVP requires validation testing be performed by CAVP

FIPS 140 Security Levels

- FIPS 140-2 is current standard; 4 security levels
- *Security level 1*: encryption algorithm is to be FIPS-approved algorithm; must be executed on production-grade equipment
  - For example, a general-purpose computer using unevaluated operating system
- *Security level 2*: requirements for security level 1, plus:
  - Physical security: tamper-evident coatings or seals or pick-resistant locks
  - Provides for role-based authentication
  - Allows software cryptography in multiuser systems when used with operating system evaluated at EAL2 or better in Common Criteria

FIPS 140 Security Levels

- *Security level 3*: requirements for security level 2 plus:
  - Enhanced physical security (available in many commercial products)
  - Identity-based authentication
  - Underlying operating system is EAL3 under specific Common Criteria PP
- Security level 4: requirements for security level 3 plus:
  - Physical security: envelope of protection around crypto module to detect. respond unauthorized attempts at physical access
  - Protection against compromise from environment
  - Software, firmware components of module can be executed on general-purpose operating system meeting EAL4 or higher

FIPS 140-2 Documentation

- Validation testing of modules uses Derived Test Requirements (DTR) for FIPS 140-2
  - Contains all vendor, certification laboratory requirements for validating module
- Implementation Guidance (IG) provides programmatic guidance of CMVP
  - Contains clarification, guidance for DTR
  - Testing, implementation guidance of Approved, non-Approved functions
  - Guidance on how validated software, firmware can be ported to similar environment and retain its validation

Impact

- Improved quality, security of cryptographic modules
- 164 modules tested by 2002, about half had security flaws; 95% had documentation errors
  - Vendors fixed these before deployment, use
- 332 cryptographic algorithms tested by 2002, about 25% had security flaws; more than 65% had documentation errors
  - Vendors fixed these before deployment, use
- By 2018, more than 1100 cryptographic modules, more than 7000 cryptographic algorithms validated

Common Criteria (CC)

- Joint project of several nations
  - US, Canada, UK, France, Germany, Netherlands, others
- Version 1.0 published in 1994
- CC *de facto* standard in US in 1998
  - TCSEC retired in 2000

Common Criteria (CC)

- Arrangement on the Recognition of the Common Criteria Certifications in the Field of Information Technology Security
  - First signed in 1998 by US, UK, France, Germany, Canada
  - Australia, New Zealand signed in 1999
  - As of 2017, 28 nations in the CCRA
- Expanded to allow nations to join as authorizing (certification producing) and/or consuming (certification recognizing) members
  - As of 2017, 17 authorizing nations including the US, UK, Australia, Canada, France, Germany

CC Methodology

- CC Documents
  - Provide overview of methodology, functional and assurance requirements, Evaluation Assurance Levels (EALs)
- CC Evaluation Methodology (CEM)
  - Provides detailed guidelines for evaluation at levels EAL1–EAL4, commonly used assurance requirements not in any EAL
  - EAL1–EAL4 are low to medium trust; EAL5–EAL7 are high assurance
- Evaluation Scheme (or National Scheme)
  - Provide infrastructure necessary to implement CC evaluations
  - Each country does this in its own way

Evaluation (National) Schemes

- CC documents, CEM set fundamental criteria, EALs, evaluation strategy
- Countries may have different methods of selecting evaluators, structuring interactions between vendors and evaluators, awarding certifications, etc.
  - Example: in US, National Institute of Standards and Technologies (NIST) implements Common Criteria Evaluation and Validation Scheme (CCEVS); NIST accredits commercial labs to do the evaluations; NIST then validates the evaluation and awards the EALs

Terms

- *TOE Security Policy* (TSP): set of rules regulating how assets are managed, protected, distributed within a system
- *TOE Security Functions* (TSF): all hardware, firmware, software of the system that must be relied on for correct enforcement of TSP
  - Generalizes concept of TCB

Evaluation of Protection Profiles (PP)

- *CC Protection Profile*: implementation-independent set of security requirements for category of systems that meet specific consumer needs
- PP has 6 sections
  - Introduction
  - Conformance claims
  - Security problem definition
  - Security objectives
  - Extended components definition
  - Security requirements

Structure of Protection Profiles (PP)

- *Introduction*; contains PP reference information, TOE overview
- *Conformance claims*: does PP claim conformance to any other PPs, packages
  - *Strict conformance*: requires evidence all PP requirements are met and ST or PP claiming conformance is instantiation of the PP while allowing ST or PP claiming conformance to be broader than itself
  - *Exact conformance*: requires ST claiming conformance use exact same security requirements (type of strict conformance)
  - *Demonstrable conformance*: requires evidence that ST/PP claiming conformance solves generic security problem described in PP

Structure of Protection Profiles (PP)

- *Security problem definition*: presents
  - *Assumptions* about intended use, environment of use
  - *Threats* to assets requiring protection; threat agents, type of attacks, assets that are targets of attacks
  - *Organizational security policies* that the product must abide by
- *Extended components definition*: defines components needed in a PP not defined in CC

Structure of Protection Profiles (PP)

- *Security objectives*: defines security objectives, rationale
  - *Security objectives for the TOE* must be traced back to identified threats, organizational policies
  - *Security objectives for the operational environment* must be traced to threats not completely countered by system, organizational policies, assumptions not met by system

Structure of Protection Profiles (PP)

- *Security requirements*: functional, assurance requirements
  - *Security functional requirements* (SFR) usually drawn from CC, or supplied by author
  - *Security assurance requirements* may be based on EAL
  - *Security requirements rationale* demonstrates requirements are traceable to, and meet, security objectives

PP-Module

- Uniquely referenced construct defining a set of elements addressing optional set of security features added to base product type
  - Must refer to at least one Base-PP providing mandatory requirements and base TOE type
  - Complements security problem definition, objectives, requirements of Base-BB by adding new elements or giving more detailed set of elements
  - Must be evaluated as part of PP-Configuration

PP-Configuration

- Composite of one or more PP-Modules with associated Base-PP
- Cannot have additional content not found in selected PP-Modules or Base-PPs
- Evaluation rules for these based on evaluation rules for standard PPs

Evaluation of System against Security Target

- First part: evaluation of ST in accordance with assurance class ASE: Security Target Evaluation
- Second part: evaluation of system against ST
- *Security target*: implementation-dependent set of security requirements and specifications to be used as basis for evaluation of identified system

Structure of the Security Target

- ST consists of 7 sections
  - ST introduction
  - Conformance claims
  - Security problem definition
  - Security objectives
  - Extended component definition
  - Security requirements
  - TOE summary specification

Structure of the Security Target

Introduction section has 4 parts

- *ST reference*: precise information used to control, identify the ST
- *TOE reference*: precise information used to control, identify system to which ST refers
- *TOE overview*: brief description of TOE acceptable as abstract for use in evaluated product lists; also states type of TOE (router, firewall, OS, etc.)
- *TOE description*: more detailed description of TOE to aid in understanding its security requirements

Structure of the Security Target

Conformance claims section has 4 parts

- *CC Conformance claims*: statement of conformance to CC
  - *Part 2 (3) conformant: uses only functional requirements from CC part 2 (3)*
  - *Part 2 (3) extended*: also uses extended requirements defined by vendor
- *PP claim*: list of PPs to which ST is conformant
- *Package claim*: identifies packages (EALs) to which ST claims conformance
  - *Conformant*: security functional, assurance requirements identical to those in package
  - Augmentation: security functional, assurance requirements of ST include all those of package plus at least 1 additional requirement

Structure of the Security Target

Conformance claims section has 4 parts

- *CC Conformance claims*: statement of conformance to CC
  - *Part 2 (3) conformant: uses only functional requirements from CC part 2 (3)*
  - *Part 2 (3) extended*: also uses extended requirements defined by vendor
- *PP claim*: list of PPs to which ST is conformant
- *Package claim*: identifies packages (EALs) to which ST claims conformance
  - Conformant: security functional, assurance requirements identical to those in package
  - Augmentation: demonstrates TOE type consistent with claimed PP, security objectives, requirements are consistent with those of claimed PP

Structure of the Security Target

Conformance claims section, 4th part

- *Conformance rationale*: show the following:
  - TOE type consistent with claimed PP
  - Security problem definition (SPD) in ST is consistent with that in claimed PP
  - Security objectives in ST are consistent with those in claimed PP
  - Security requirements in ST are consistent with those in claimed PP

Structure of the Security Target

Security Problem Definition: includes

- *Assumptions* about intended usage, environment of use
- *Threats* to assets requiring protection in terms of threat agents, types of attacks, targets
- *Organizational security policies* that the system must respect

Structure of the Security Target

Security Objectives: two types of objectives

- *Security objectives for the TOE* must be traced back to aspects of identified threats, organizational policies
- *Security objectives for the operational environment* must be traced back to threats, assumptions, organizational policies not completely met or countered by system
- Security objectives rationale shows security objectives counter threat, meet assumptions, enforce organizational security policy

Structure of the Security Target

Extended components definition defines components in ST not defined in CC Parts 2 and 3

- New definitions must be modeled after existing CC Part 2 components

Structure of the Security Target

Security Requirements cover functional, assurance requirements

- *Security functional requirements* drawn from CC Part 2
  - If none appropriate, ST author can supply others
- *Security assurance requirements* drawn from CC Part 3, may be based on an EAL
  - Author may add extra security assurance requirements from CC or may supply others, including security requirements for environment
- *Security requirements rationale* shows requirements for system, environment traceable to and meet objectives
- *Justification* for any security requirement dependencies not satisfied

Structure of the Security Target

TOE Summary Specification defines instantiation of system security requirements

- High-level description of how TOE meets claimed security functionsl requirements
- High-level description of how TOE protects itself from interference, logical tampering, bypass

CC Requirements

- Requirements divided into classes based on common purposes
- Classes broken into families
- Families made up of components
  - Definitions of detailed requirements, dependent requirements, definition of hierarchy of requirements
- Functional requirements
- Assurance requirements
  - EALs built from these

CC Security Functional Requirements

11 classes, each with at least 1 family; family has:

- Management section with specific information about management issues for subdivisions, requirements of family
- Audit section identifies auditable events
- Hierarchical dependencies
  - Requirement A hierarchical to requirement B if A's functional requirements offer more security, or is more restrictive, than those of B
- Nonhierarchical dependencies also identified
  - May cross classes

CC Security Functional Requirements Classes

- FAU: Security Audit; 6 families
  - Audit automatic response, data generation, analysis, review, storage
- FCO: Communication; 2 families
  - Nonrepudiation of origin, receipt
- FCS: Cryptographic Support; 2 families
  - Cryptographic key management, operation
- FDP: User Data Protection, 13 families
  - Access control policies, information flow policies; object reuse, data authentication, rollback, stored data integrity

CC Security Functional Requirements Classes

- FIA: Identification and Authentication; 6 families
  - Authentication failures, definitions of user attributes, user/subject binding
- FMT: Security Management; 7 families
  - Security attribute and management of TSF and TSF data, roles, attribute expiration
- FPR: Privacy; 4 families
  - Anonymity, pseudonymity, unlinkability, unobservability
- FPT: Protection of Security Functions, 14 families
  - TSF physical protection, trusted recovery, confidentiality, integrity, availability of exported TSF data, replay detection, TSF self-tests

CC Security Functional Requirements Classes

- FRU: Resource Utilization; 3 families
  - Fault tolerance, resource allocation, priority of service
- FTA: TOE Access; 6 families
  - limitations on multiple concurrent sessions, session locking and termination, TOE access history, access banners, system entry constraints
- FTP: Trusted Path; 2 families
  - Inter-TSF channel function, trusted path

Example

- Class FAU has 6 families
  - For each family, management section identifies management functions of class FMT that should be considered
  - For each family, audit section identifies auditable events that must be addressed if component FAU_GEN is selected in the PP or ST
- Component FAU_SSA: security audit analysis; 4 components in it
  - FAU_SSA.1, potential violation analysis component not hierarchical to any other component
  - FAU_SSA.1 depends on requirement FAU_GEN.1 (from another FAU family); so if FAU_SSA.1 selected, also must select FAU_GEN.1

Example

- FAU_SSA.1 has 2 functional requirements
- FAU_SSA.2 has 2 functional requirements
  - It is profile-based anomaly detection
  - Hierarchical to FAU_SSA.1, meaning requirements of FAU_SSA.2 more stringent than those of FAU_SSA.1, subsuming requirements
  - FAU_SSA.2 depends on FIA_UID.1,, requirement for family in another class

CC Security Assurance Requirements Classes

- APR: Protection Profile Evaluation; 6 families
  - One for each section of PP
- ACE: Protection Profile Configuration Evaluation; 8 families
  - Used to evaluate a PP-Configuration
- ASE: Security Target Evaluation; 7 families
  - One for each section of ST
- ADV: Development; 6 families
  - Security architecture, functional specification, implementation representation, TSF internals, TOE design, security policy modeling

CC Security Assurance Requirements Classes

- AGD: Guidance Documentation; 2 families
  - Operational user guidance, preparative procedures
- ALC: Life Cycle; 7 families
  - configuration management capabilities and scope, delivery, development security, flaw remediation, tools and techniques, life cycle definition
- ATE: Tests; 4 families
  - Test coverage, depth, functional tests, independent testing
- AVA: Vulnerabilities Assessment; 1 family
- ACO: Composition; 5 families
  - Composition rationale, development evidence, reliance of dependent component, composed TOE testing, composition vulnerability analysis

Evaluation Assurance Levels (EALs)

- EAL1: *Functionally Tested*; level based on analysis of security functions using functional and interface specifications, and examination of provided guidance documentation
  - Requires unique TOE identification
  - Applicable to systems for which you need some confidence in correct operations, but security threats are not serious
- EAL2: *Structurally Tested*; EAL1 + analysis of basic description of TOE architecture
  - Supported by search for vulnerabilities, evidence of developer testing, and vulnerability analysis to show resistance to basic attacks
  - Applicable to systems requiring low to moderate level of assurance, but complete development record might not be available

Evaluation Assurance Levels (EALs)

- EAL3: *Methodically Tested and Checked*; like EAL2, but security function analysis requires architectural description of TOE design
  - Supported as EAL2, and high-level design for developer to base testing upon, use of development environment controls, configuration management
- EAL4: *Methodically Designed, Tested, and Reviewed*; EAL3 + low-level design, complete interface description, basic modular TOE design, subset of implementation to inputs for security function analysis
  - Supported as EAL3, and implementation representation, vulnerability analysis to show resistance to enhanced-basic attacks
  - Applicable to systems requiring moderate to high level of assurance
  - Likely to be highest EAL level for retrofitting existing product line

Evaluation Assurance Levels (EALs)

- EAL5: *Semiformally Designed and Tested*; like EAL4, but add modular TSF design and full implementation to input for security functional analysis; requires semiformal functional specification, modular high-level design; comprehensive configuration management
  - Applicable to systems requiring high level of assurance
  - Highest EAL level for rigorous commercial development practices supported by moderate amount of computer security engineering
- EAL6: *Semiformally Verified Design and Tested*; like EAL5, but add formal model of security policies, semiformal TOE design and functional specification; methodical vulnerability search to address penetration attackers with high potential
  - Applicable to systems in high-risk situations where protected assets valuable enough to justify cost, effort of development, certification

Evaluation Assurance Levels (EALs)

- EAL7: *Formally Designed and Tested*; formal presentation of functional specification, high-level design; implementation representation used as basis for testing
  - Complete confirmation of developer test results, independent of developer
  - Applicable to systems in extremely high-risk situations, requires substantial security engineering

Comparison of Levels of Trust

TCSEC	ITSEC	CC	Other
D	E0	*no equivalent*	
no equivalent	*no equivalent*	EAL1	Private lab testing
C1	E1	EAL2	OS for FIPS 140-2 L2
C2	E2	EAL3	OS for FIPS 140-2 L3
B1	E3	EAL4	OS for FIPS 140-2 L4
B2	E4	EAL5	
B3	E5	EAL6	
A1	E6	EAL7	

Evaluation Process (in the U.S.)

- Controlled by CC-Evaluation Methodology (CEM) and NIST
  - Evaluations by NIST-accredited commercial labs for fee
- Labs may offer support for vendors preparing for evaluation
  - But staff that do this cannot work as evaluators on the evaluation
- Vendor chooses accredited laboratory
  - Work with vendor to develop baseline schedule
  - Co-ordinate with validating body
- When done, evaluation findings go to validating agency

Evaluation Process (in the U.S.)

- Evaluation of product or system
  - Some schemes require completed ST that has passed all CEM units *before* they agree to evaluate a product or system
  - Other schemes require an ST that is mostly complete
- Evaluation schemes
  - US scheme only accepts evaluations against NIST-approved PP
  - Other schemes may accept evaluations claiming EAL level
- Lab sends evaluation findings to national validating agency
  - This agency determines whether to accept evaluation and award EAL rating

SOG-IS International Cooperation Agreement

- Senior Officials Group Information Systems Security (SOG-IS) agreement: mutual recognition agreement between participating government organizations, agencies in EU or European Free Trade Association (EFTA)
  - Originally signed in 1997
  - Updated in 1999 to incorporate CC

SOG-IS International Cooperation Agreement

- Modified in 2010 to include authorizing (certification producing) and/or consuming (certification recognizing) participation
  - As of 2017, 8 authorizing nations including the UK, France, Germany; 6 consuming participants including Austria, Finland, Poland
  - Recognition of levels above EAL4 limited to approved technical areas
- 2 levels of certificate producers
  - Recognition of CC certificates claiming EAL1–EAL4
  - Recognition of CC certificates at higher levels for defined technical areas with SOG-IS approved scheme for that level

SOG-IS International Cooperation Agreement

- Participants collaborate to:
  - Standardize Common Criteria PPs and certificate policies among CC schemes in Europe
  - Present common position within the CCRA
  - Develop PPs when EU Commission issues IT security-related directive
- Authorizing nations still perform EAL3, EAL4 evaluations
- Two technical areas covered by SOG-IS; PPs developed for products in these areas
  - Smartcards, similar devices like passports
  - Hardware devices with security boxes

Common Criteria Users Forum

- Common Criteria Users Forum (CCUF): international group composed of people from academia, consultants, users, governments, CC laboratories, vendors, etc.
- Promotes worldwide recognition of:
  - CC evaluations
  - Focused technical communities to develop cPPs
  - Policies, processes for maintaining evaluation on future versions of product
  - Policies, processes for evaluating systems composed of evaluated products
- Governance: CCUF Management Board

CC Discussion

- Much more complete than functional requirements of previous evaluation technologies
- PP or ST may not be as strong as TCSEC classes
  - Fewer experts have reviewed it
  - Not yet faced test of time
- Some CC requirements derived from requirements of previous methodologies
  - These may have more credibility than other requirements
- Ultimately correctness of ST is up to vendor, evaluation team

CC Discussion

- CC project board manages interpretations
  - Any national scheme can submit their interpretations
  - Final interpretations become required on all subsequent evaluations
  - “Criteria creep”: newer evaluations may have to meet more stringent requirements than older evaluations
- Evaluation process monitored by validating body

CC Discussion

- CC documentation, methodology are evolving
  - 8 official versions of CC/CEM so far
  - New technical committees form, continue to develop cPPs
- Common Criteria Management Board (CCMB): international body that maintains CC, ensures CCRA is operated as defined by its rules
  - Each signatory of CCRA has representative on CCMB
  - CCMB discusses change proposals from CCRA participants; determination I Agreed (worthy of being adopted internationally by CC), Concurred (proposal acceptable, does not violate mutual recognition, but not worthy of international adoption), Disagreed (proposal violates mutual recognition rules or too incomplete to be accepted)

System Security Engineering Capability Maturity Model (SSE-CMM)

- Methodology for developing secure systems
  - Based on Software Engineering Capability Maturity Model (SE-CMM)
  - It focuses on processes used to develop system
- Provides maturity levels
  - Contrast with previous ones, which provide trust levels
- Can provide assurance evidence, thereby increasing confidence in trustworthiness of product
- Organized into processes, maturity levels

SSE-CMM Model: Processes

- *Process capability*: range of expected results that can be achieved by following process
  - Indicates potential; a predictor of future project outcomes
- *Process performance*: measure of actual results achieved
- *Process maturity*: extent to which process is explicitly defined, managed, measured, controlled, effective

SSE-CMM Model: Processes

11 systems security engineering process areas:

- Administrator security controls
- Assess impact
- Assess security risk
- Assess threat
- Assess vulnerability
- Build assurance argument
- Coordinate security
- Monitor system security posture
- Provide security input
- Specify security needs
- Verify and validate security

Definition of each process area contains goal, set of supporting base practices (total of 61 base practices within all areas)

Example: Assess Threat Process Area

- Goal: threats to security of system be identified, characterized
- Base processes:
  - Identify Natural Threats
  - Identify Human-Made Threats
  - Identify Threat Units of Measure
  - Assess Threat Agent Capability
  - Assess Threat Likelihood
  - Monitor Threats and Their Characteristics

SSE-CMM Model: Project Practices

11 process areas for project, organizational practices (adapted from SE-CMM):

- Ensure quality
- Manage configuration
- Manage project risk
- Monitor and control technical effort
- Plan technical effort
- Define organization's system engineering process
- Improve organization's system engineering process
- Manage product line evolution
- Manage systems enfineering support environment
- Provide ongoing skills and Knowledge
- Coordinate with suppliers

SSE-CMM: Capability Maturity Levels

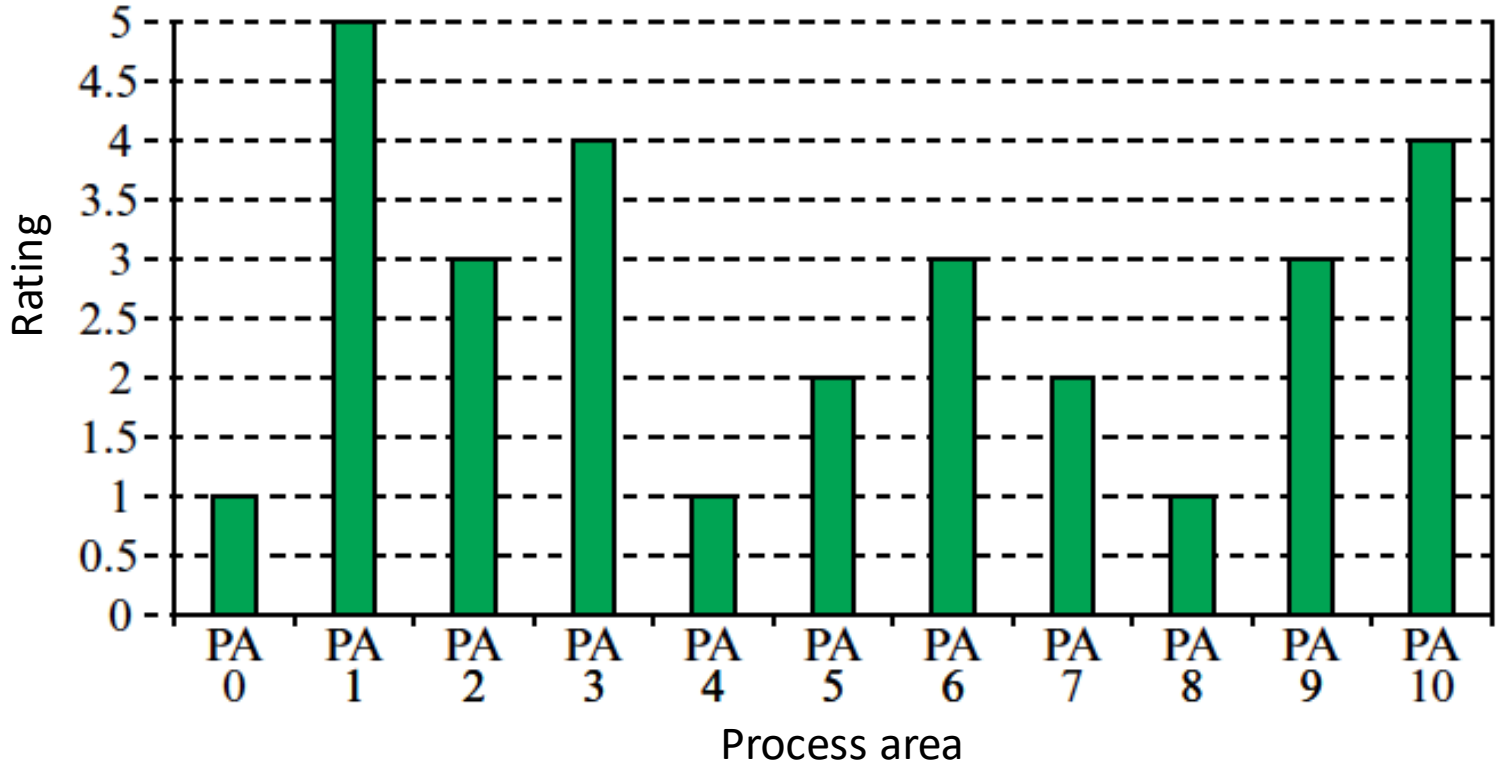
- *Performed informally*: base processes are performed
- *Planned and tracked*: project-level definition, planning, performance verification issues addressed
- *Well-defined*: defining, refining standard practice, coordinating it across organization
- *Quantitatively controlled*: establishing measurable quality goals, objectively managing their performance
- *Continuously improving*: improve organizational capability and process effectiveness

Using SSE-CMM

- Straightforward analysis of existing processes
  - See which base processes have been met
  - See which maturity levels achieved
- Pick project area
- Identify area goals, base processes defined for that process area
  - If all present, assess processes against capability maturity levels
- Result is identification of current level of maturity for each base process in process area
- Repeat for each process area; level of maturity is lowest level represented by set of levels of base process

Rating Profile

- Tabular representation of process areas vs. maturity levels



Key Points

- First public, widely used evaluation technique was TCSEC
  - Led to much research, development of other approaches addressing concerns about TCSEC
- Other methodologies:
  - ITSEC in Europe, CTCPEC in Canada, FC in US
- These led to Common Criteria international evaluation methodology
- Other current evaluation techniques
  - FIPS 140-2 (cryptographic modules, managed by US NIST, Canadian CSE)
  - SSE-CMM (process oriented)