L17: Assurance

Hui Chen, Ph.D.
Dept. of Engineering & Computer Science
Virginia State University
Petersburg, VA 23806

Acknowledgement

- Many slides are from or are revised from the slides of the author of the textbook
  - Matt Bishop, Introduction to Computer Security, Addison-Wesley Professional, October, 2004, ISBN-13: 978-0-321-24774-5. Introduction to Computer Security @ VSU's Safari Book Online subscription
  - http://nob.cs.ucdavis.edu/book/book-intro/slides/

Outline

- Trust
- Problems from lack of assurance
- Types of assurance
- Life cycle and assurance
- Waterfall life cycle model
- Other life cycle models
- Adding security afterwards

“Secure” or “Trusted”?

- Two terms
  - Secure systems
  - Trusted systems
- Secure systems
  - Can we build an “absolutely” secure system?
  - Ultimate, albeit unachievable goal
- Trusted systems
  - Trust: a belief or desire that a computer entity will do what it should to protect resources and be safe from attack
  - Trust: has very *specific* meaning in computer security

Trust

- Definition
  - An entity is *Trustworthy* if there is sufficient credible evidence leading one to believe that the system will meet a set of requirements
  - *Trust* is a measure of trustworthiness relying on the evidence

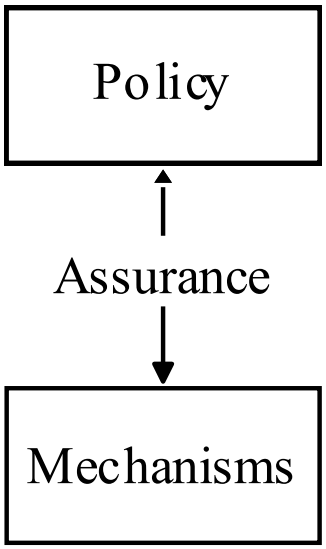
Assurance

- Definition
  - *Security Assurance*, or simply *assurance*, is confidence that an entity meets its security requirements based on evidence provided by applying assurance techniques

Example Assurance Techniques

- Formal vs. informal
- Development methodology
  - Examples
    - Brief description of the methodology to be followed
    - System Security Engineering Capability Maturity Model (SSE-CMM)
- Formal methods for design analysis
  - Use machine-parsable languages with tools
  - Formal mathematical proof
- Testing

Assurance, Policy, and Mechanisms



Statement of requirements that explicitly defines the security expectations of the mechanism(s)

Provides justification that the mechanism meets policy through assurance evidence and approvals based on evidence

Executable entities that are designed and implemented to meet the requirements of the policy

Information Assurance vs. Security Assurance

- Information assurance
  - Refers to the ability to access information and preserve the quality and security of that information
- Differs from security assurances
  - Information assurance focuses on the threats to the information and mechanisms used to protect information
  - But not on the correctness, consistency, or completeness of the requirements and implementation of those mechanisms

Trusted System

- Definition
  - A trusted system is a system that has been shown to meet well-defined requirements under an evaluation by a credible body of experts who are certified to assign trust ratings to evaluate products and systems

Example Evaluation Criteria

- Two earlier ones
  - Trusted Computer System Evaluation Criteria
  - Information Technology Security Evaluation Criteria
- Replaced by
  - Common Criteria
- These mythologies provide a “level of trust”
  - Each level has more stringent requirements than the previous one
- Experts evaluate and review the evidence of assurance

Problem Sources

1. Requirements definitions, omissions, and mistakes
2. System design flaws
3. Hardware implementation flaws, such as wiring and chip flaws
4. Software implementation errors, program bugs, and compiler bugs
5. System use and operation errors and inadvertent mistakes
6. Willful system misuse
7. Hardware, communication, or other equipment malfunction
8. Environmental problems, natural causes, and acts of God
9. Evolution, maintenance, faulty upgrades, and decommissions

Examples

- Challenger explosion
  - Sensors removed from booster rockets to meet accelerated launch schedule
- Deaths from faulty radiation therapy system
  - Hardware safety interlock removed
  - Flaws in software design
- Bell V22 Osprey crashes
  - Failure to correct for malfunctioning components; two faulty ones could outvote a third
- Intel 486 chip
  - Bug in trigonometric functions

Role of Requirements

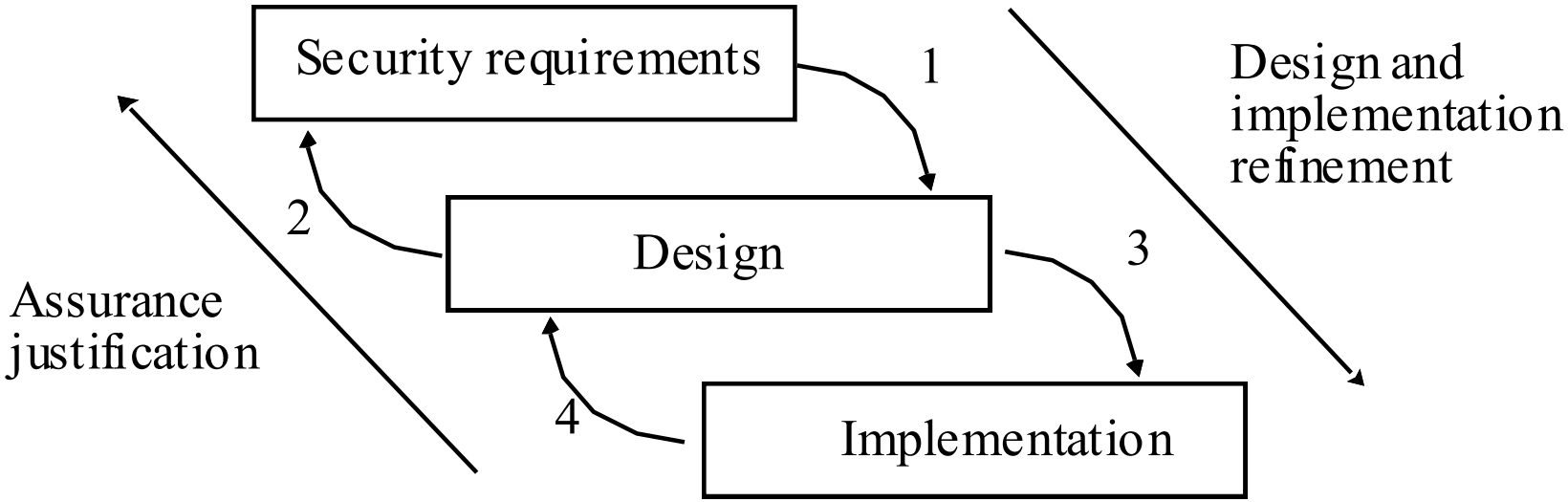
- *Requirements* are statements of goals that must be met
  - Vary from high-level, generic issues to low-level, concrete issues
- *Security objectives* are high-level security issues
- *Security requirements* are specific, concrete issues

Type of Assurance

- *Policy assurance* is evidence establishing security requirements in policy is complete, consistent, technically sound
- *Design assurance* is evidence establishing design sufficient to meet requirements of security policy
- *Implementation assurance* is evidence establishing implementation consistent with security requirements of security policy
- *Operational assurance* is evidence establishing system sustains the security policy requirements during installation, configuration, and day-to-day operation
  - Also called *administrative assurance*

Life Cycle

- Conception
- Manufacture
- Deployment
- Fielded Product Life



Conception

- Idea
  - Decisions to pursue it
- Proof of concept
  - See if idea has merit
- High-level requirements analysis
  - What does “secure” mean for this concept?
  - Is it possible for this concept to meet this meaning of security?
  - Is the organization willing to support the additional resources required to make this concept meet this meaning of security?

Manufacture

- Develop detailed plans for each group involved
  - May depend on use; internal product requires no sales
- Implement the plans to create entity
  - Includes decisions whether to proceed, for example due to market needs

Deployment

- Delivery
  - Assure that correct masters are delivered to production and protected
  - Distribute to customers, sales organizations
- Installation and configuration
  - Ensure product works appropriately for specific environment into which it is installed
  - Service people know security procedures

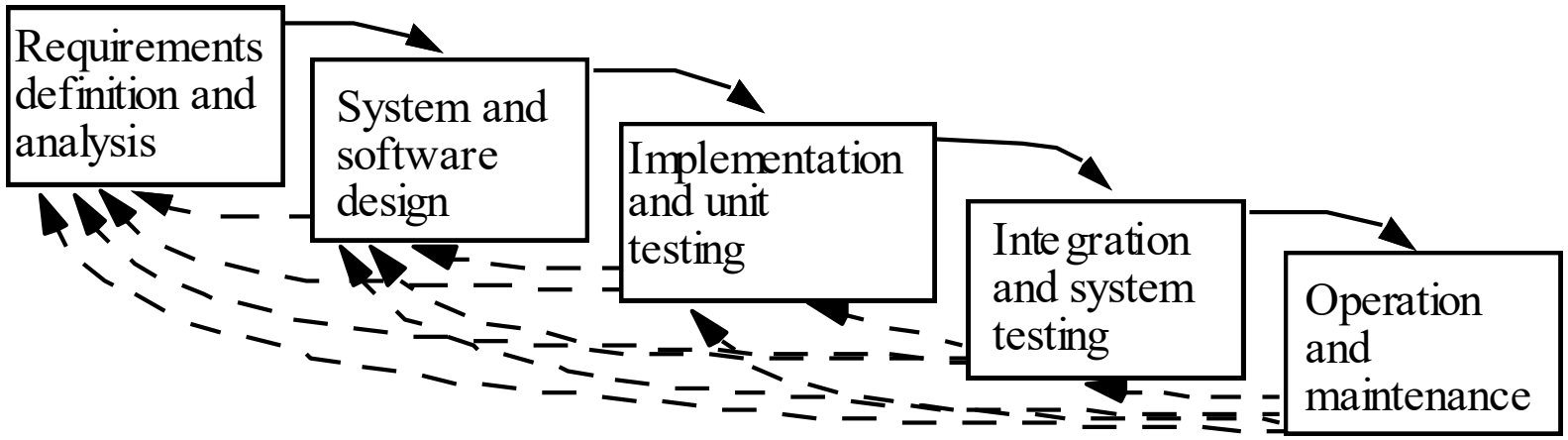
Fielded Product Life

- Routine maintenance, patching
  - Responsibility of engineering in small organizations
  - Responsibility may be in different group than one that manufactures product
- Customer service, support organizations
- Retirement or decommission of product

Waterfall Life Cycle Model

- Requirements definition and analysis
  - Functional and non-functional
  - General (for customer), specifications
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

Relationship of Stages



Other Models of Software Development

- Exploratory programming
- Prototyping
- Formal transformation
- System assembly from reusable components
- Extreme programming

Exploratory Programming

- Develop working system quickly
- Used when detailed requirements specification cannot be formulated in advance, and adequacy is goal
- No requirements or design specification, so low assurance

Prototyping

- Objective is to establish system requirements
- Future iterations (after first) allow assurance techniques

Formal Transformation

- Create formal specification
- Translate it into program using correctness-preserving transformations
- Very conducive to assurance methods

System Assembly

- System assembly from reusable components
  - Depends on whether components are trusted
  - Must assure connections, composition as well
  - Very complex, difficult to assure

Extreme Programming

- Rapid prototyping and "best practices"
- Project driven by business decisions
- Requirements open until project complete
- Programmers work in teams
- Components tested, integrated several times a day
- Objective is to get system into production as quickly as possible, then enhance it
- Evidence adduced *after* development needed for assurance

Security: Built In or Add On?

- Think of security as you do performance
  - You do not build a system, then add in performance later
    - Can "tweak" system to improve performance a little
    - Much more effective to change fundamental algorithms, design
- You need to design it in
  - Otherwise, system lacks fundamental and structural concepts for high assurance

Reference Validation Mechanism

- *Reference monitor* is access control concept of an abstract machine that mediates all accesses to objects by subjects
- *Reference validation mechanism* (RVM) is an implementation of the reference monitor concept.
  - Tamperproof
  - Complete (always invoked and can never be bypassed)
  - Simple (small enough to be subject to analysis and testing, the completeness of which can be assured)
    - Last engenders trust by providing assurance of correctness

Examples

- *Security kernel* combines hardware and software to implement reference monitor
- *Trusted computing base* (*TCB*) is all protection mechanisms within a system responsible for enforcing security policy
  - Includes hardware and software
  - Generalizes notion of security kernel

Adding On Security

- Key to problem: analysis and testing
- Designing in mechanisms allow assurance at all levels
  - Too many features adds complexity, complicates analysis
- Adding in mechanisms makes assurance hard
  - Gap in abstraction from requirements to design may prevent complete requirements testing
  - May be spread throughout system (analysis hard)
  - Assurance may be limited to test results

Example

- 2 AT&T products
  - Add mandatory controls to UNIX system
  - SV/MLS
    - Add MAC to UNIX System V Release 3.2
  - SVR4.1ES
    - Re-architect UNIX system to support MAC

Comparison: Architecture

- Architecting of System
  - SV/MLS: used existing kernel modular structure; no implementation of least privilege
  - SVR4.1ES: restructured kernel to make it highly modular and incorporated least privilege

Comparison: File Attributes

- File Attributes (*inodes*)
  - SV/MLS added separate table for MAC labels, DAC permissions
    - UNIX inodes have no space for labels; pointer to table added
    - Problem: 2 accesses needed to check permissions
    - Problem: possible inconsistency when permissions changed
    - Corrupted table causes corrupted permissions
  - SVR4.1ES defined new inode structure
    - Included MAC labels
    - Only 1 access needed to check permissions

Summary

- Assurance is critical for determining trustworthiness of systems
- Different levels of assurance, from informal evidence to rigorous mathematical evidence
- Assurance needed at all stages of system life cycle
- Building security in is more effective than adding it later