ECS 235B, Lecture 10

January 30, 2019

Controversy

- McLean:
  - "value of the BST is much overrated since there is a great deal more to security than it captures. Further, what is captured by the BST is so trivial that it is hard to imagine a realistic security model for which it does not hold."
  - Basis: given assumptions known to be non-secure, BST can prove a non-secure system to be secure

†-Property

- State (b, m, f, h) satisfies the †-property iff for each $s \in S$ the following hold:
  1. $b(s\text{: }\underline{\text{a}}) \neq \varnothing \Rightarrow [\forall o \in b(s\text{: }\underline{\text{a}})\ [\ f_c(s)\ dom\ f_o(o)\ ]\ ]$
  2. $b(s\text{: }\underline{\text{w}}) \neq \varnothing \Rightarrow [\forall o \in b(s\text{: }\underline{\text{w}})\ [\ f_o(o) = f_c(s)\ ]\ ]$
  3. $b(s\text{: }\underline{\text{r}}) \neq \varnothing \Rightarrow [\forall o \in b(s\text{: }\underline{\text{r}})\ [\ f_c(s)\ dom\ f_o(o)\ ]\ ]$
- Idea: for writing, subject dominates object; for reading, subject also dominates object
- Differs from *-property in that the mandatory condition for writing is reversed
  - For *-property, it's object dominates subject

Analogues

The following two theorems can be proved

- $\Sigma(R, D, W, z_0)$ satisfies the †-property relative to $S' \subseteq S$ for any secure state z_0 iff for every action $(r, d, (b, m, f, h), (b', m', f', h'))$, W satisfies the following for every $s \in S'$
  - Every $(s, o, p) \in b - b'$ satisfies the †-property relative to S'
  - Every $(s, o, p) \in b'$ that does not satisfy the †-property relative to S' is not in b
- $\Sigma(R, D, W, z_0)$ is a secure system if z_0 is a secure state and W satisfies the conditions for the simple security condition, the †-property, and the ds-property.

Problem

- This system is *clearly* non-secure!
  - Information flows from higher to lower because of the †-property

Discussion

- Role of Basic Security Theorem is to demonstrate that rules preserve security
- Key question: what is security?
    - Bell-LaPadula defines it in terms of 3 properties (simple security condition, *-property, discretionary security property)
    - Theorems are assertions about these properties
    - Rules describe changes to a *particular* system instantiating the model
    - Showing system is secure requires proving rules preserve these 3 properties

Rules and Model

- Nature of rules is irrelevant to model
- Model treats “security” as axiomatic
- Policy defines “security”
    - This instantiates the model
    - Policy reflects the requirements of the systems
- McLean’s definition differs from Bell-LaPadula
    - … and is not suitable for a confidentiality policy
- Analysts cannot prove “security” definition is appropriate through the model

System Z

- System supporting weak tranquility
- On *any* request, system downgrades *all* subjects and objects to lowest level and adds the requested access permission
  - Let initial state satisfy all 3 properties
  - Successive states also satisfy all 3 properties
- Clearly not secure
  - On first request, everyone can read everything

Reformulation of Secure Action

- Given state that satisfies the 3 properties, the action transforms the system into a state that satisfies these properties and eliminates any accesses present in the transformed state that would violate the property in the initial state, then the action is secure
- BST holds with these modified versions of the 3 properties

Reconsider System Z

- Initial state:
    - subject s, object o
    - C = {High, Low}, K = {All}
- Take:
    - $f_c(s)$ = (Low, {All}), f_o(o) = (High, {All})
    - $m[s, o] = \{ \underline{\text{w}} \}$, and $b = \{ (s, o, \underline{\text{w}}) \}$.
- s requests $\underline{\text{r}}$ access to o
- Now:
    - $f'_o(o)$ = (Low, {All})
    - $(s, o, \underline{\text{r}}) \in b'$, $m'[s, o] = \{\underline{\text{r}}, \underline{\text{w}}\}$

Non-Secure System Z

- As $(s, o, \underline{r}) \in b' - b$ and $f_o(o)$ *dom* $f_c(s)$, access added that was illegal in previous state
    - Under the new version of the Basic Security Theorem, System Z is not secure
    - Under the old version of the Basic Security Theorem, as $f'_c(s) = f'_o(o)$, System Z is secure

Response: What Is Modeling?

- Two types of models
  1. Abstract physical phenomenon to fundamental properties
  2. Begin with axioms and construct a structure to examine the effects of those axioms
- Bell-LaPadula Model developed as a model in the first sense
  - McLean assumes it was developed as a model in the second sense

Reconciling System Z

- Different definitions of security create different results
  - Under one (original definition in Bell-LaPadula Model), System Z is secure
  - Under other (McLean’s definition), System Z is not secure

Key Points

- Confidentiality models restrict flow of information
- Bell-LaPadula models multilevel security
  - Cornerstone of much work in computer security
- Controversy over meaning of security
  - Different definitions produce different results

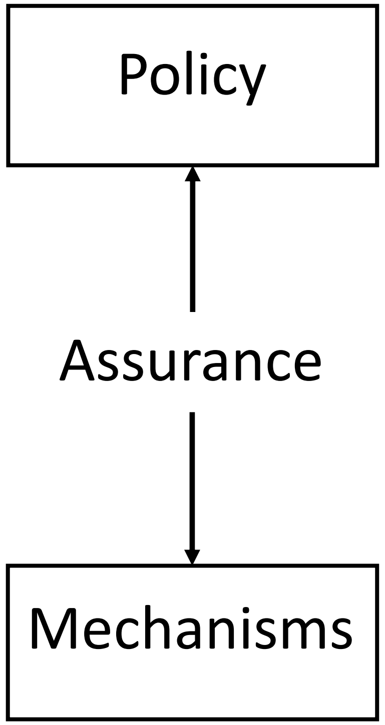
Introduction to Assurance

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

Trust

- *Trustworthy* entity has 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* is confidence that an entity meets its security requirements based on evidence provided by applying assurance techniques

Relationships



Statement of requirements that explicitly define 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 imple-mented to meet the requirements of the policy

Trusted System

- 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 or assurance levels to evaluated products and systems
  - Use specific methodologies to gather assurance evidence
  - These methodologies typically have increasing ”levels of trust”

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 satisfied
  - Vary from high-level, generic issues to low-level, concrete issues
- *Security objectives* are high-level security issues
- *Security requirements* are specific, concrete issues
- *Security policy* is set of specific statements that, when enforced, result in a secure system
  - Alternatively, a statement that partitions states of system into a set of authorized states and a set of unauthorized states
- *Security model* describes a family of policies, systems, or entities and is more abstract than a policy
  - A policy is specific to particular entities

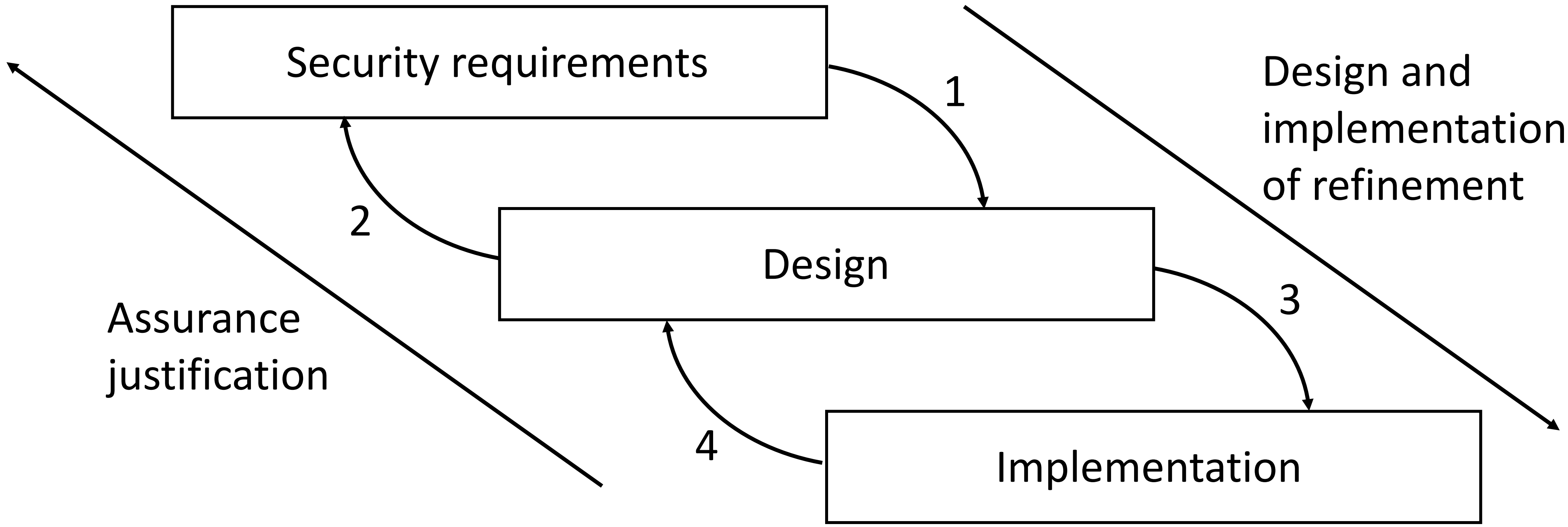
Types of Assurance

- *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

Types of Assurance

- *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


Security requirements
1
Design and implementation of refinement
2
Design
3
Assurance justification
4
Implementation

Life Cycle for Building Secure, Trusted Systems

- Life cycle process establish discipline, control in the building of a product or system
  - This provides confidence in consistency, quality of resulting system
- Assurance *requires* life cycle model end engineering process in *every* situation
  - Size and complexity will vary
- Life cycle defined in stages

Generic Life Cycle Model

These are present in all models, but the emphasis and focus is different for each project, and will be more detailed than what is presented here

- 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?
- Identify threats, assumptions

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
  - Software development, engineering process is in this stage

Deployment

- Delivery
    - Assure that correct masters are delivered to production and protected
    - Assure integrity of what is delivered to customers, sales organizations
- Installation and configuration
    - Ensure product works appropriately for specific environment into which it is installed
    - Service people know security procedures
- Example of configuration failure
    - 2013: Target breached via a third party vendor, as network architected with improper security controls

Fielded Product Life

- Routine maintenance, patching
  - Responsibility of engineering in small organizations
  - Responsibility may be in different group than one that manufactures product
  - Example of failure: 2017 Equifax breach believed due to failing to install an important system patch, resulting in breach of financial information for hundreds of millions of people
- Customer service, support organizations
- Retirement or decommission of product