Rigorous and Traceable Decomposition and Recomposition of Security Requirements

Leo Marcus

Trusted Computer Systems Department

The Aerospace Corporation

Motivation

- Gain assurance that system-level security requirements are satisfied by implemented system
- Enable system development where assumptions made by one segment match the expectations of other segments
- Creation of Common Criteria Security Targets with traceability to system-level security requirements
  - NIST and NSA program to evaluate IT product conformance to international standards: National Information Assurance Partnership Common Criteria and Evaluation and Validation Scheme for IT Security (NIAP CCEVS)

Goals of Presentation

- Describe a method
  - to derive system-wide security requirements
  - allocate them to system components
  - to an arbitrary level of detail
  - in a uniform manner

What is not included

- How to calculate residual security risk
- How to evaluate requirements’ cost/benefit
- Acquisition strategy
- … Among others
- These other considerations involve elaboration, not destruction, of the model.

Essential Quotables

- "In theory there is no difference between theory and practice, but in practice there is." (Yogi Berra)
  - Need the theory to guide the practice
  - And vice versa
- "Make things as simple as possible, but not simpler." (Albert Einstein)
  - The method is commonsense -- almost obvious -- but not entirely!

Benefits

- Assurance
  - That the system meets its (security) requirements
- Traceability
  - From requirement to source and vice versa
- Maintainability
  - Changes and upgrades can be made with confidence
- Composability
  - Consistency at the interface between different groups working on different components of the same system
- Applicability to the Common Criteria

Compromises

- 100% assurance a requirement is completely met is not realistic
- Cost/benefit/risk analysis determines countermeasure/requirement
  - Applies to
    - allocation to subcomponents
    - derivation and refinement within a component
    - suitability of product at unit implementation level

Main Security Concerns

- Confidentiality
  - Information is not divulged in an unauthorized manner
- Integrity
  - Information is not altered in an unauthorized manner
- Availability
  - System resources are not degraded or denied in an unauthorized manner

Secondary Security Concerns

- Authentication
  - The true requester of access to information or resources is the same as the apparent requester
- Non-repudiation
  - The receiver of information cannot deny receipt
  - The user of a resource cannot deny that use
  - The sender of information cannot deny sending
  - The grantor of a resource cannot deny granting

Origins of Security Requirements

- Security requirements derive from three sources:
  - Threats
    - what they can do ***to*** you
  - Policy
    - what they ***tell*** you to do
  - Assumptions
    - what they will do ***for*** you

“They” is anybody or anything outside of the control of “you” (the system being designed and built)

Threats

- ***Threats*** are instances of any potential security breaches, disruptions to the secure operation of the system
- Threats can be carried out by ***attacks*** on system ***vulnerabilities*** by
    - Identified, motivated, funded adversary (“validated threat”)
    - Hypothetical adversary
    - Nature/accident
- Threats can vary greatly in severity and likelihood
- Requirements derived from threats are ***countermeasures***
    - Prevention
    - Detection
    - Response

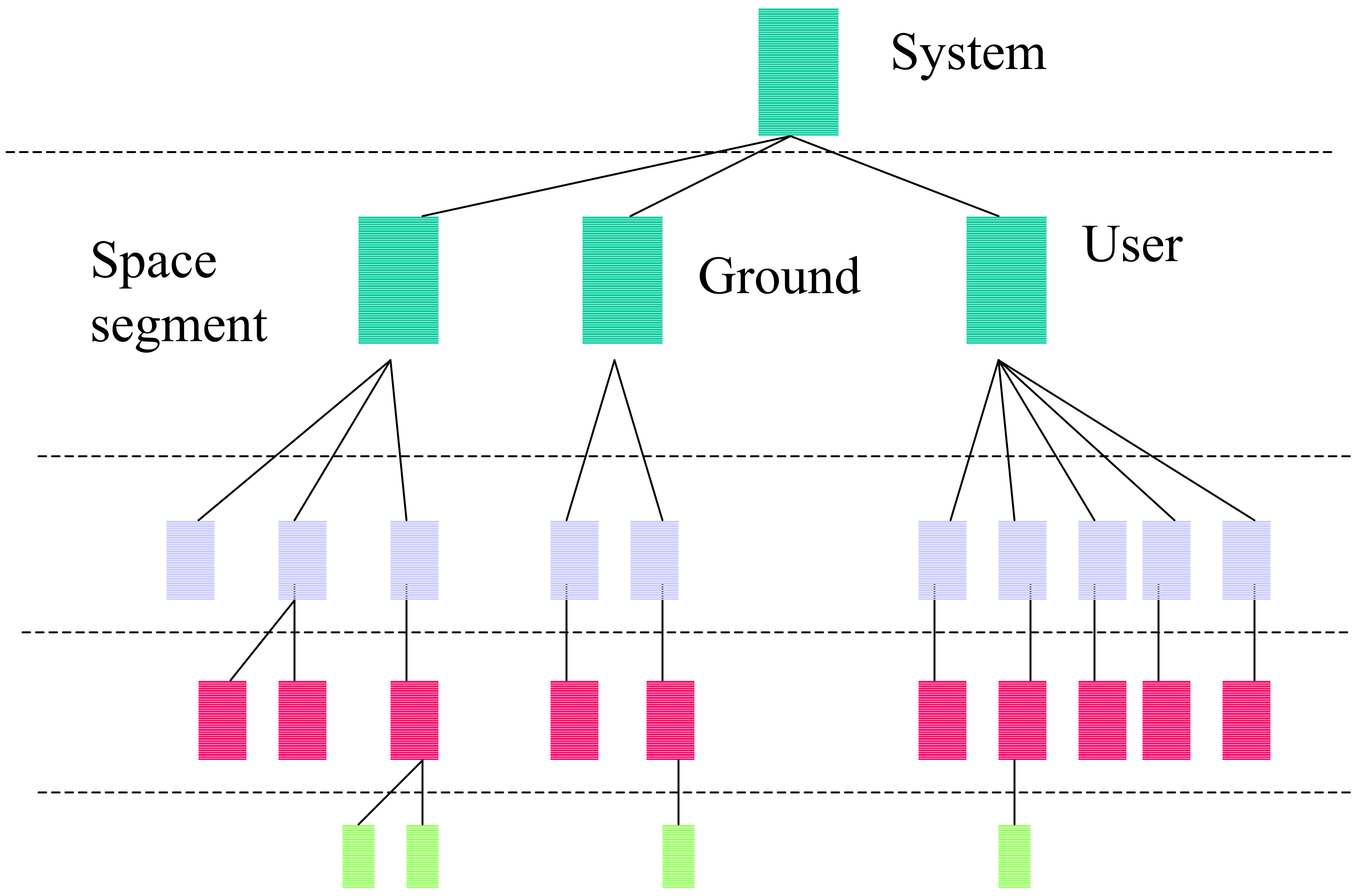
Policy

- Any security relevant directives, objectives, or design decisions that are deemed necessary aspects of the system by any organizations authorized to impose their mandates, without need for traceability to system-specific threats.
  - E.g., “Thou shalt use crypto.”
  - Could include information about the priorities of dealing with various potential threats
  - Could also include legacy material, or system increments
- Requirements derived from policy are refinements and elaborations of those policy statements appropriate to the given level of abstraction and functionality of the system component under consideration.

Assumptions

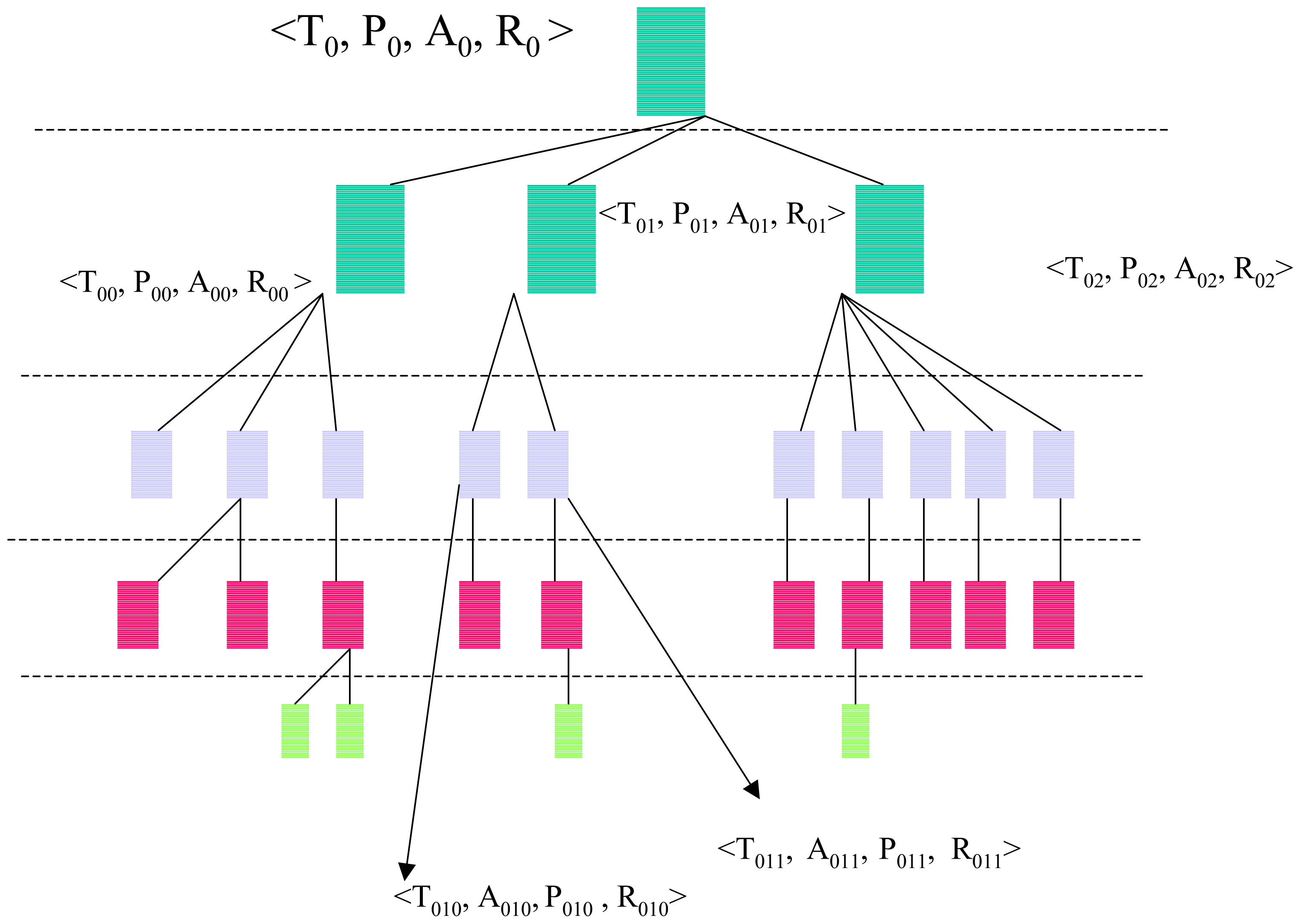
- Assumptions are properties contributing to system security, whose truth/implementation is dependent on agents/factors outside of the system being designed/built.
- Requirements derived from threats and policy may be omitted from the system being built if they can be shown to follow from the assumptions (perhaps in conjunction with other requirements.)
- Assumptions that one system component makes about its environment may need to be translated into requirements on other system components.

SYSTEM DECOMPOSITION TREE



The Datatype (T, P, A, R)

- ***Requirement Derivation***: At a given system node the connection between threats T, policies P, assumptions A, and the derived requirements R can be specified in semi-rigorous fashion.
- ***Requirement Allocation***: If the datatype (T, P, A, R) is instantiated at all nodes of a system decomposition tree, the allocations of requirements to system components can be justified in semi-rigorous fashion.



Deriving Requirements

- T, P, A ➔ R
  - Every threat in T must have a countermeasure in R
  - Every policy in P must have an elaboration in R
  - Unless an assumption in A already takes care of it

Or Else!

- If a threat T is not dealt with by a requirement, then there may be system vulnerability that can be exploited in an attack.
  - This could be a decision based on C/B/R analysis
- If a policy statement P is not covered by a requirement, then some authority is not going to be happy
- If a requirement is not traceable to any T or P, then there is a potential waste of money

Component Threat Allocation

- $T_X \rightarrow T_{X0}, T_{X1}, \ldots, T_{Xn}$
  - Each can be done independently
  - And independently of all P, A, and R
- Every system-wide threat may have a local specific interpretation at a component node level
  - Example: System-wide threat:
    - "Information may be divulged to an unauthorized party";
  - Specific interpretation:
    - Space, ground, and user have different authorized parties, and deal with different kinds of information.
- Threat data at every node should include all local interpretations of parent system threats.

Component Policy Allocation

- $P_X \rightarrow P_{X0}, P_{X1}, \ldots, P_{Xn}$
  - The P_{Xi} may be dependent on one another
  - Can be done independently of all T, A, and R
- Every system-wide policy statement may have a more specific interpretation at a given child component node
  - Example: System-wide policy:
    - "Security-relevant events must be audited";
  - Specific interpretation:
    - Space, user, and control have different security-relevant events and different means of auditing.
- Every system-wide policy statement must have a more specific interpretation at some child component node
- Policy data at every node should include all local interpretations of parent system policies.

Component Assumption Allocation

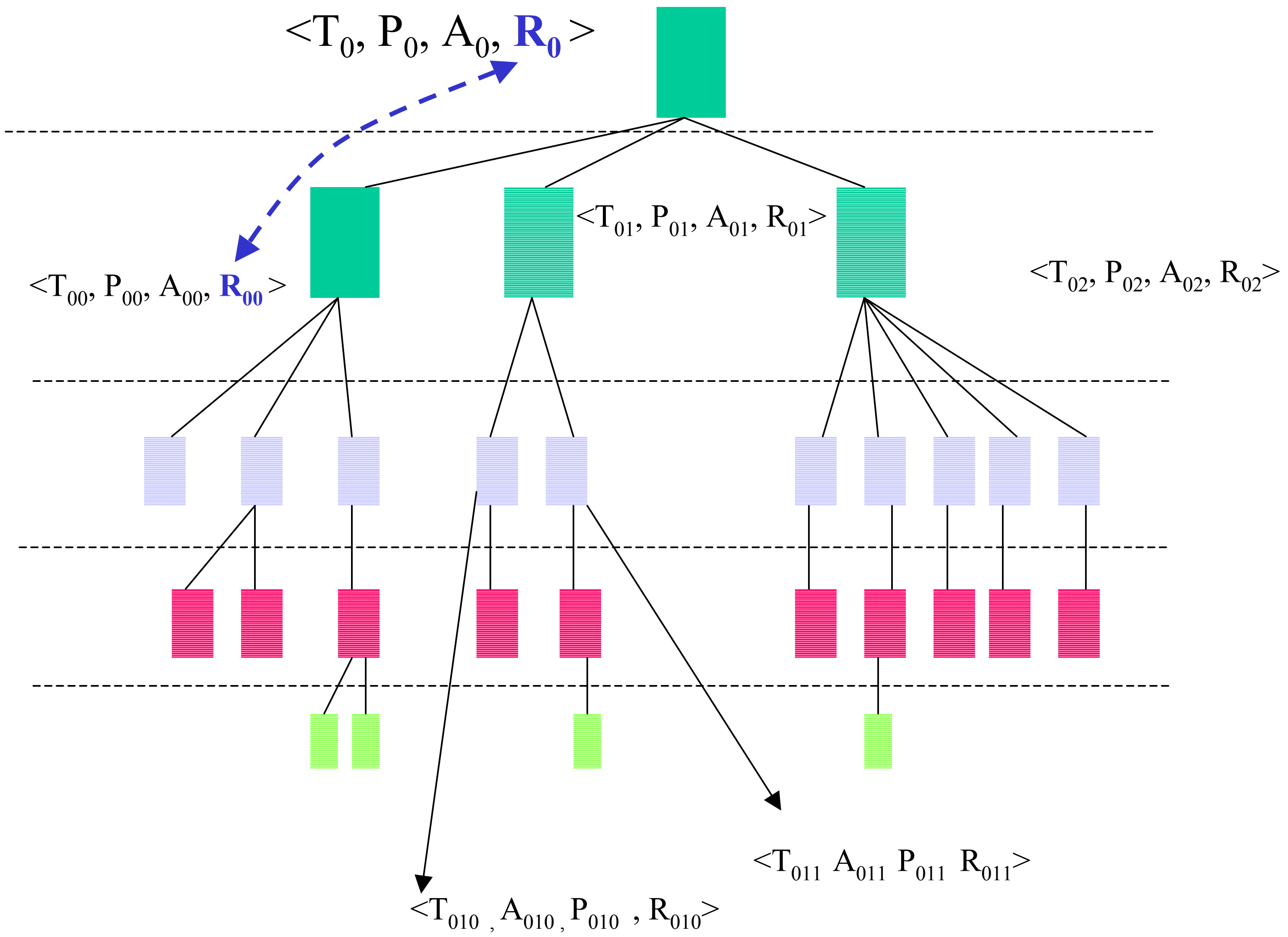
- $A_X \rightarrow A_{X0}, A_{X1}, \ldots, A_{Xn}$
  - The A_{Xi} are independent
  - Can be dependent on P and R
- Every system-wide assumption may have a more specific interpretation at a child component node
  - Example: System-wide assumption:
    - "Cryptographically authenticated messages received by the system are secure"
  - Specific interpretation:
    - Space, user, and control may receive different kinds of cryptographically authenticated messages.
- Assumptions at every node should include all local interpretations of parent system assumptions.
- Any additional assumptions made by that node have to be implied by requirements at sibling nodes.

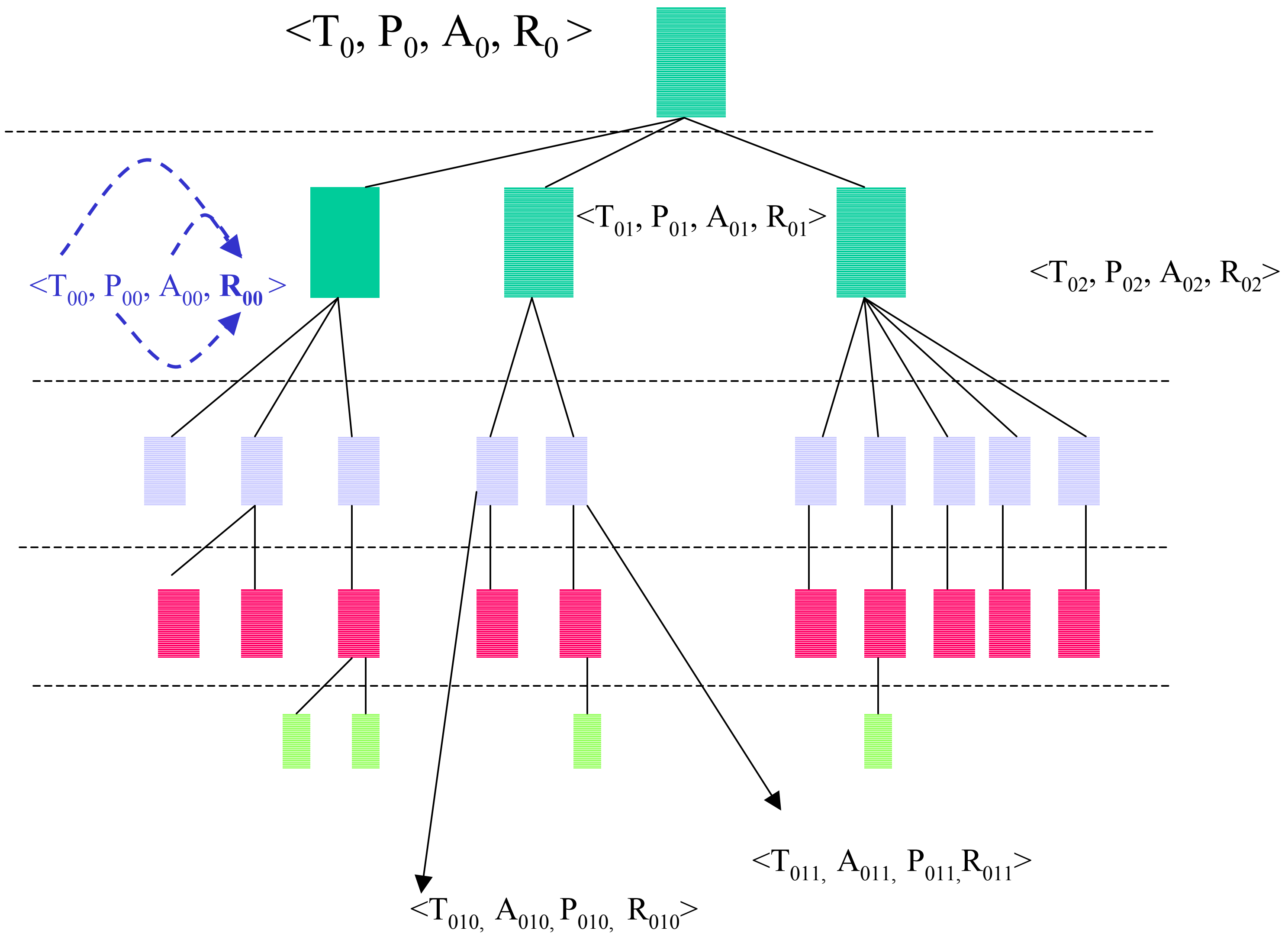
Component Requirement Allocation

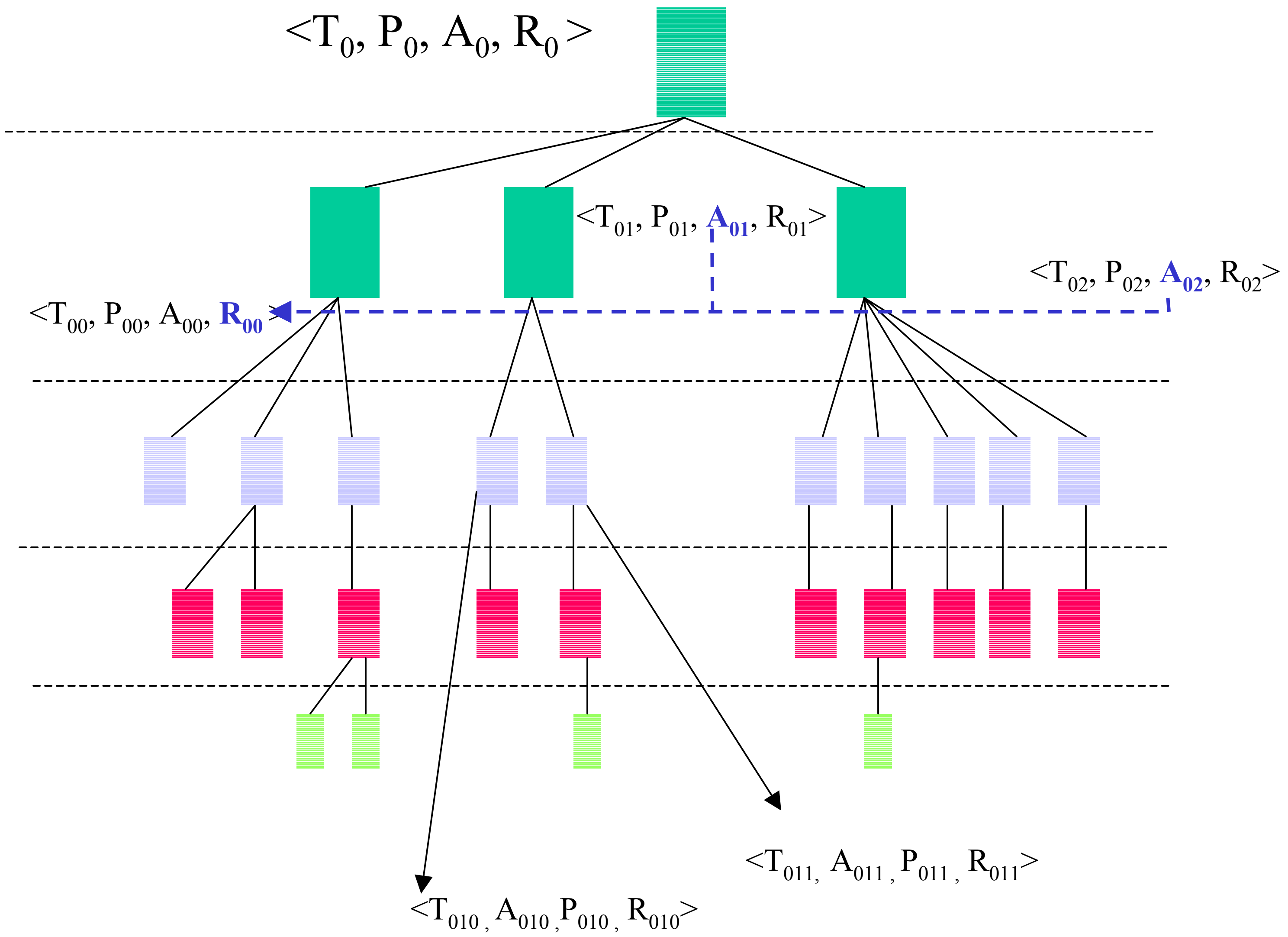
- $R_X \rightarrow R_{X0}, R_{X1}, \ldots, R_{Xn}$
  - The R_{Xi} may be dependent on one another
  - And are dependent on T, P, and A
- Every system-wide requirement may have a more specific interpretation at a given daughter component node
- Every system-wide requirement must have a more specific interpretation at some daughter component node
- Requirements at every node should include all local interpretations of parent system requirements for that node.
- Each requirement at a given node must be implied by the set of requirements at daughter nodes.

The Determination of Requirements

- The R of (T,P,A, R) is triply determined by allocation and derivation
  - Parent system node requirements have local interpretations/implementations at daughter component nodes
  - Component node requirements must also correspond to their local T, P, A data
  - Requirement might also be needed to fulfill a sibling's assumption
  - Same requirement may or may not satisfy multiple needs
- Any requirement can be replaced by an assumption if requirements at sibling nodes imply that assumption.







The Top-Down Process

- When does the process cease?
  - When the system owner decides it should; there is no objective criterion.
  - I.e., system owner does not want to impose any additional "implementation details" on the requirements at the lowest component nodes ("leaf nodes").
- New system owner of each leaf component node may start the process anew.
- Until, finally, someone actually has to do the work of implementation!

NB: Assumptions

- Importance of correctly handling assumptions typically underestimated
- Assumptions *help* to justify security
  - The “other kind of assumptions” about how dangerous the environment is should properly be classified as threats or attacks
- Assumptions in a given node can be
  - Derived from assumptions at parent node
  - Or matched with requirements at sibling nodes

Example of Assumption Derivation and Creation

- If system-level assumption is: *the confidentiality of all incoming messages that have been encrypted in an authorized manner is guaranteed*, then there are inherited assumptions only in the components (daughter nodes) that have interfaces that connect the component to something outside the system.
- The component system may also need an assumption about the confidentiality of encrypted messages incoming from other component parts of the system (sibling nodes). This assumption is reflected in requirements at those sibling nodes.

In Actuality…

- ***Process*** may not reflect the top-down nature of the finished ***product.***
- May depend on nature of system being designed:
  - Are components *created* to implement system design
  - Or are components "users" whom the system will serve
- "Users" at system component nodes may generate their own T, P, A, or R, which need to be fedback into system decomposition paradigm.
- Allocation requires negotiation between siblings
  - Parent requirement can be achieved by different allocations to daughter components.
  - Who gets the requirement and who gets the assumption

Check your work!

- Recompose to make sure system-level requirements are adequately met by composition of component requirements
- There are two generic sources of error:
  - Interaction of countermeasures may generate new vulnerabilities
  - Given that a node's derived component requirements may not be completely satisfied, we have lost the guarantee that the top-level system security requirements will be completely satisfied in the implemented system.
    - Cost/benefit trades at each level can compound the discrepancy between the top-level requirements and the lowest-level requirements, which will be implemented.
    - Assume that each subsystem requirement is partially, but "adequately", met by the composition of daughter requirements. However, those daughter requirements are only partially met by their daughters, etc. So, starting at the bottom level, if there are gaps at each level, the requirement error is compounded to an unknown degree by the time the recomposition reaches the top.

Connection to Common Criteria

- Product-level Security Targets exist at leaf level components (lowest level).
- The required sections on threats, policies, assumptions, and “objectives” (what we have been calling “requirements”) are ready, willing, and correct!

Summary

When it comes to system security:

- Keeping track explicitly of
  - Threats
  - Policies
  - Assumptions
  - Requirements
- At every node in an architectural decomposition
- Is a good thing for assurance, maintainability, consistency, and Common Criteria