Boundary Flow Modeling (BFM)

Security Policy, Architecture, and Behavior Modeling for Distributed Systems

Dr. Richard Neely, CISSP
rbn@marzen.com

Märzen Group LLC
http://www.marzen.com/

ACSAC 26, Layered Assurance Workshop
December 2010

The Message

Boundary Flow Modeling describes security characteristics in terms of data flow histories at element boundaries.

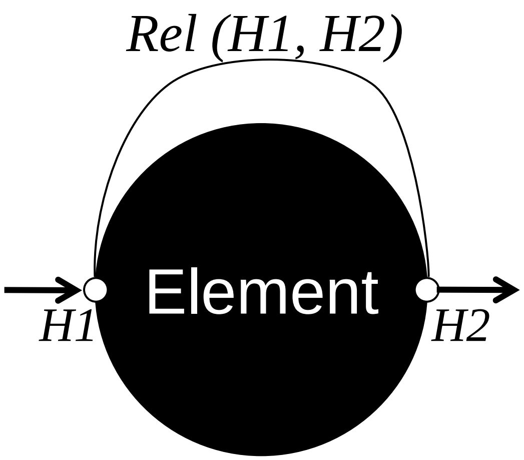
We have shown this method to be effective for distributed systems.

Contents

- Boundary Flow Modeling, Briefly
- Some Identified Security Modeling Needs + Characteristics of a Solution
- BFM Methods and Examples
- How BFM Meets the Identified Needs
- Current BFM Evolutionary Developments

BFM, Briefly

- Characteristics that BFM models
  - Policy (security requirements)
  - Architecture (high level design)
  - Behavior of elements (system, subsystems, components), viewed as black boxes
- The key of understanding BFM
  - The "words" in the modeling "language" are: histories of data flows across external interfaces of elements
  - The "sentences" are: logical relationships among histories
  - The "stories" are: inferences among relationships



Identified Security Modeling Needs

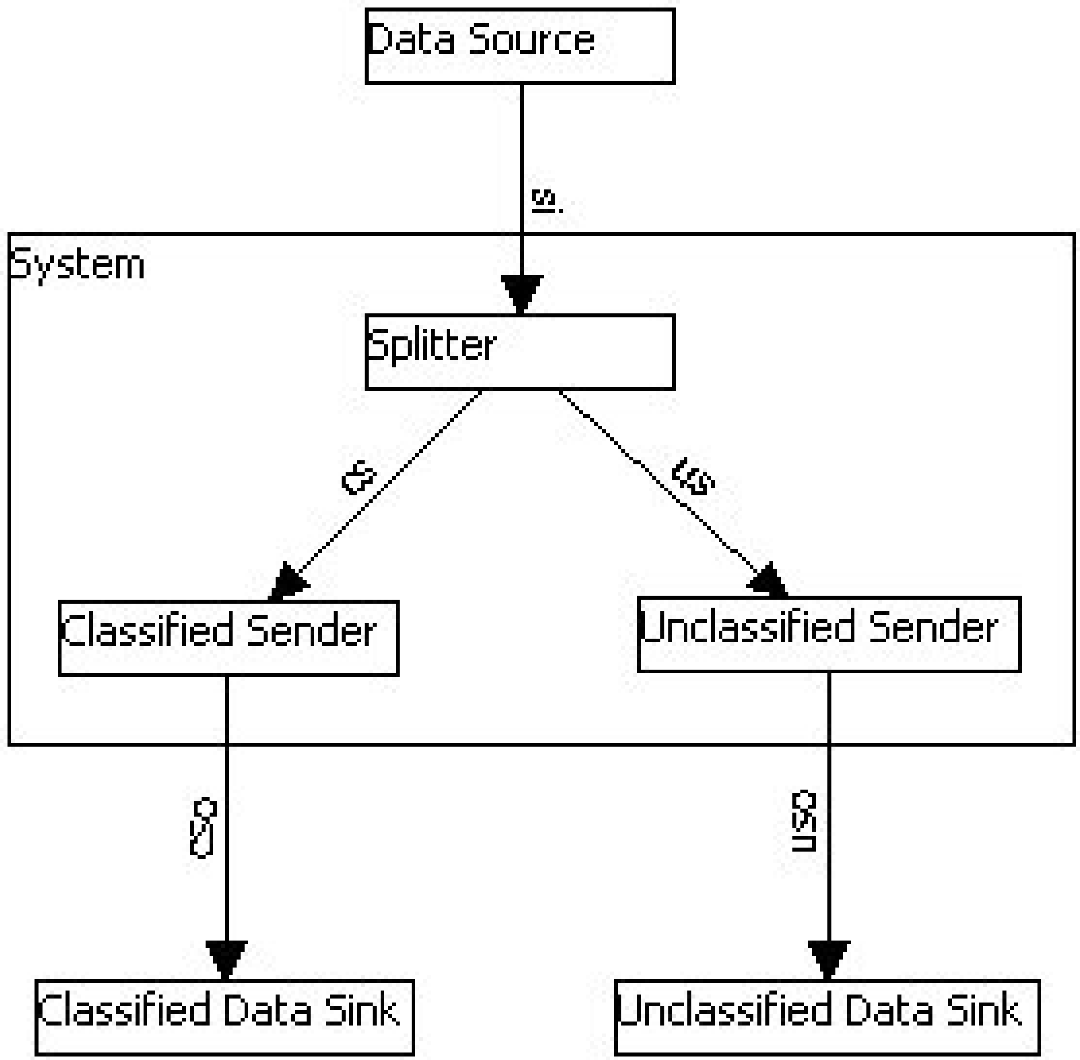
- Primary need:
  Modeling policy, architecture, and behavior
  of distributed systems
- Related need:
  Addressing the security composition problem
- Solution characteristics—a solution would have to:
  - Provide a black box view of individual targets.
  - Provide an alternative to state modeling—
    models need to be in terms of external boundaries.
- We propose that BFM is such a solution.

Märzen Group

Detailing the BFM Process

- Define an example:
  Data Sorter, a simple (distributed) system
- Walk through the process:
  Perform the steps of the process on the example.
- Identify “real” examples:
  Systems we addressed with BFM

A Simple Example: Data Sorter

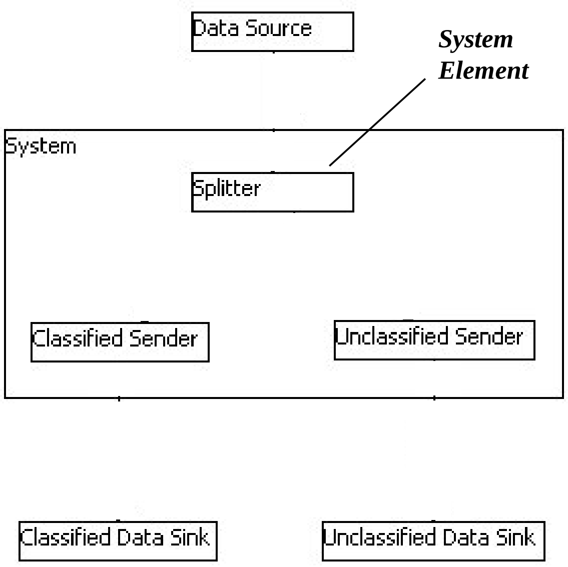


The Process

Diagram of Phases

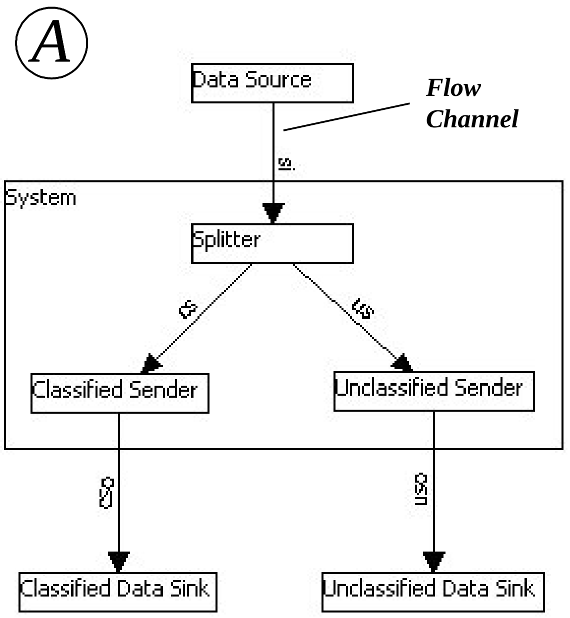
Phase 1

Security Architecture
Structure Modeling...



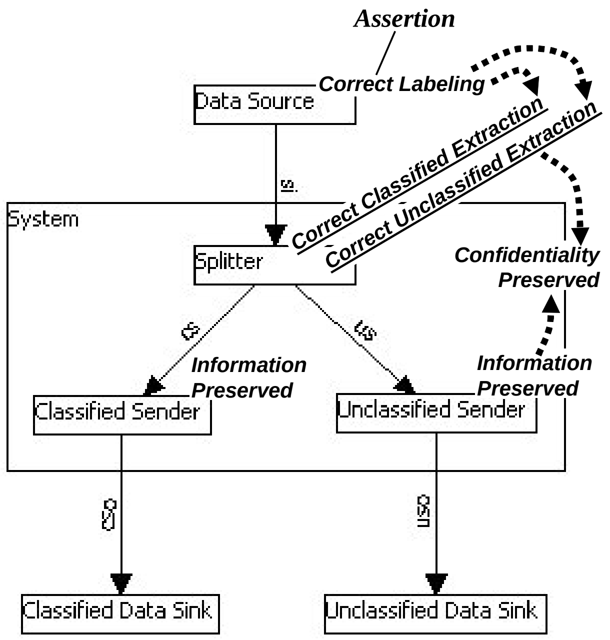
Phase 2

...Enhanced by
Information Flow...



Phases 3 & 4

...with Chain-of-Logic
Assertion Dependencies



Märzen Group

Phase Details

- *Phase 1*: Express the architecture
  Elements and component relationships
- *Phase 2*: Interfaces and data flows
  Expressed in terms of histories at interfaces
- *Phase 3*: Security constraints/behavior
  Expressed as relationships among histories
- *Phase 4*: Inferences among relationships
  Element assumptions in terms of assertions of component and peer elements
- *Phase 5*: Chain of Logic
  Applying *modus ponens* to the inferences: Validating system policy from leaf elements

Märzen Group

“Real” Examples—Actual Systems

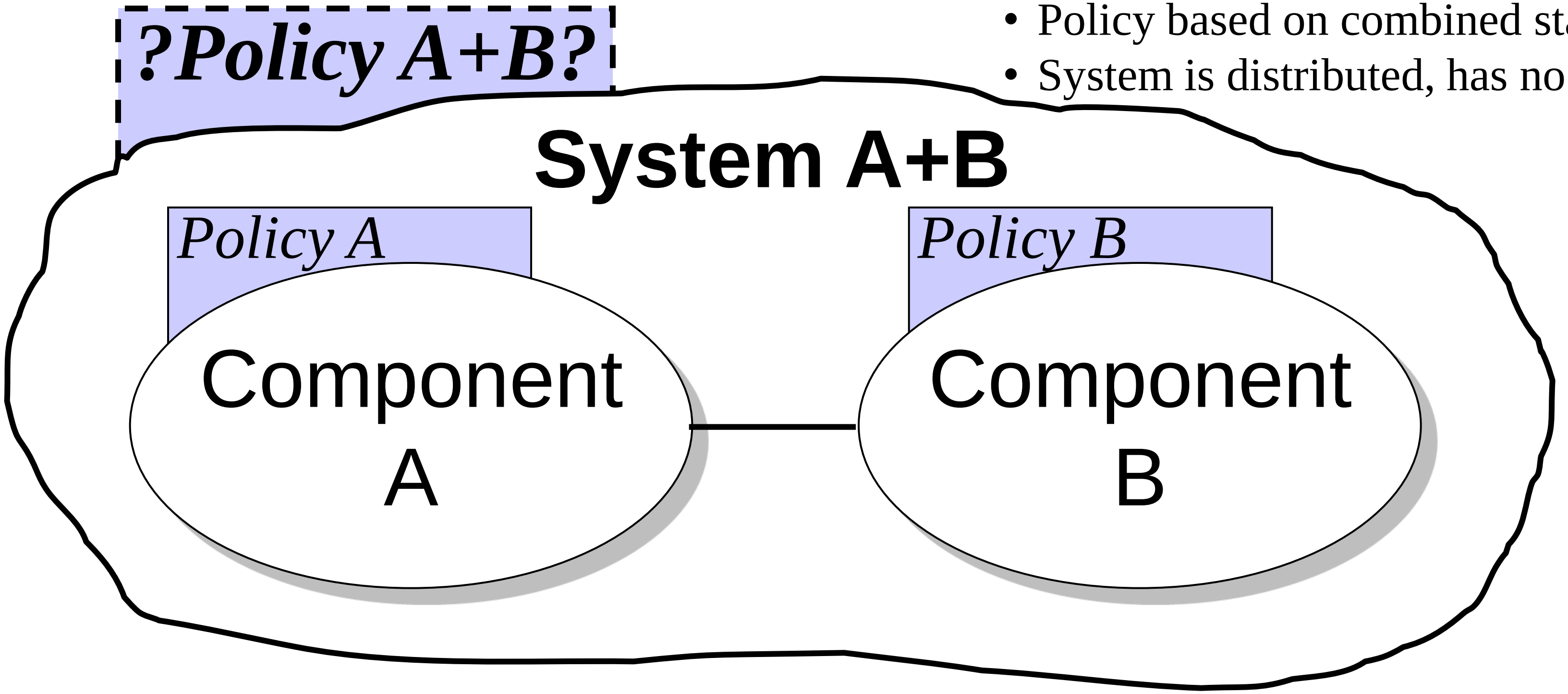
- Multinet Gateway and network environment
  - MLS network gateway (RADC and NSA)—1985-1990
- File Server example
  - Formal design modeling to validate Gypsy environment (Current Endorsed Tools List Example—National Computer Security Center (NCSC)—1991)
- F-22A Weapon System architecture and platforms
  - (Air Force—1992-1999)
- Joint Simulation System (JSIMS): Warfighter Training System
  - Two-enclave modeling and simulation system (joint sponsorship—1999-2001)

Does BFM Meet the Identified Needs?

- Solution characteristics
  - Statement: black box view of element modeling
  - Statement: modeling in terms of interfaces, not state
  - Conclusion: BFM has these characteristics
- Primary need
  - BFM models policy, architecture, and behavior of elements.
  - BFM is appropriate for distributed systems. (permits nondeterminism)
- Related need
  - BFM approach addresses the composition problem.

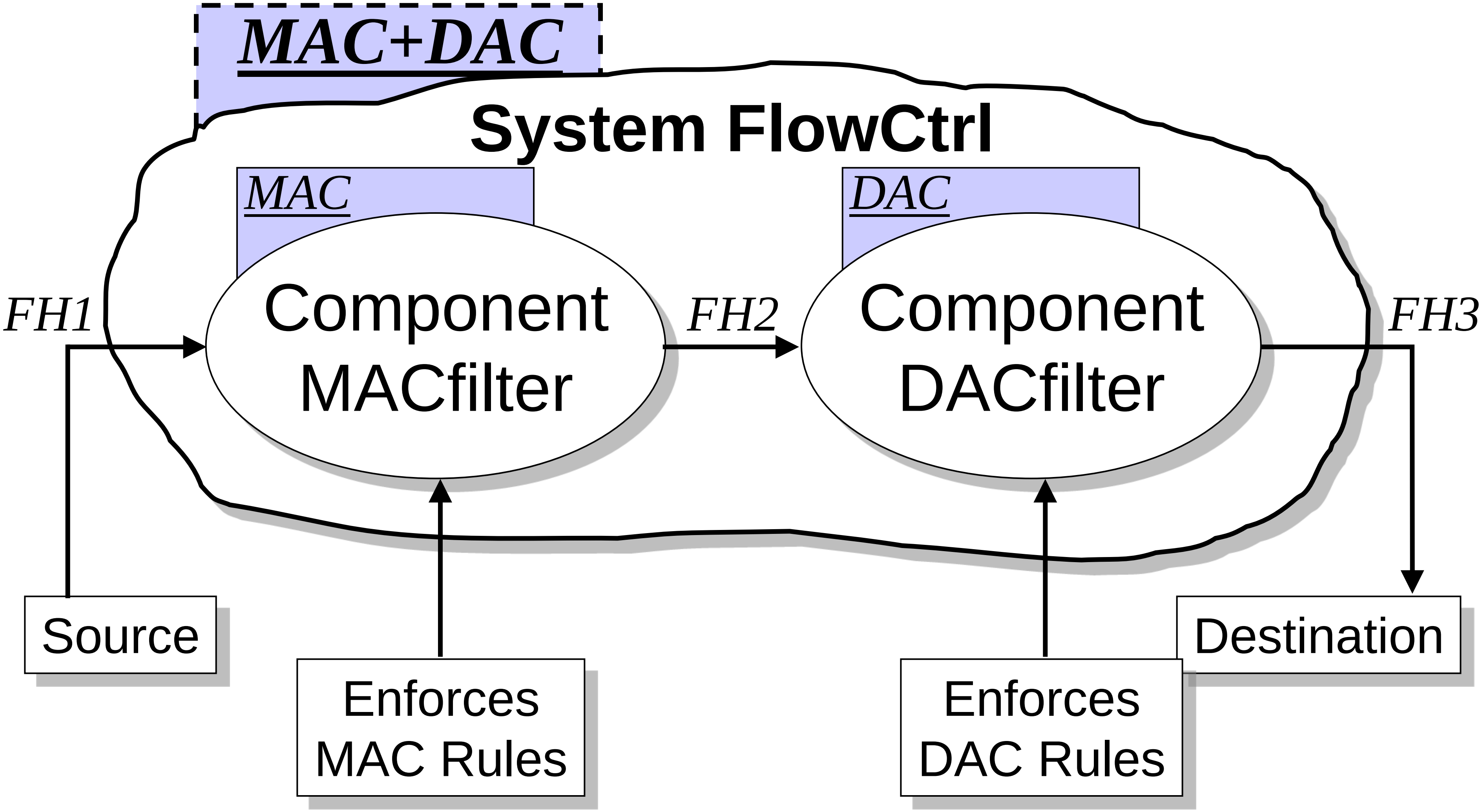
A Statement of the Security Composition Problem

- Policy based on combined state, *or*
- System is distributed, has no state



"If you have two components, each with a security policy, what is the policy enforced (if any) when the two components are combined?"

Making Sense of Security Composition



Composition Example Policies in BFM

- *MAC* Policy:
  Every packet in *FH2* has the same content as a packet in *FH1* with the MAC rules satisfied.
- *DAC* Policy:
  Every packet in *FH3* has the same content as a packet in *FH2* with the DAC rules satisfied.
- *MAC+DAC* Policy:
  Every packet in *FH3* has the same content as a packet in *FH1* with both the DAC rules and the MAC rules satisfied.
- To demonstrate based on system architecture:
  MAC Policy **AND** *DAC* Policy
  IMPLIES *MAC+DAC* Policy

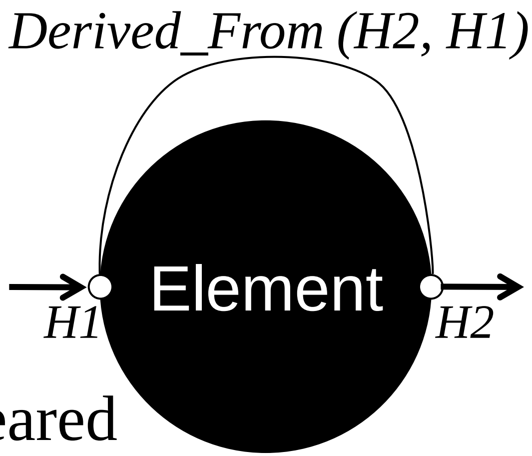
Märzen Group

Current BFM Evolutionary Development

- Soundness of flow history relationships
  - Issue of logical soundness of flow history relationships for separated elements
- Integrating BFM and state models
  - Value and approach of model integration within distributed systems
- Tool support for BFM
  - Need, past attempts, and plans

Soundness of Flow History Relationships

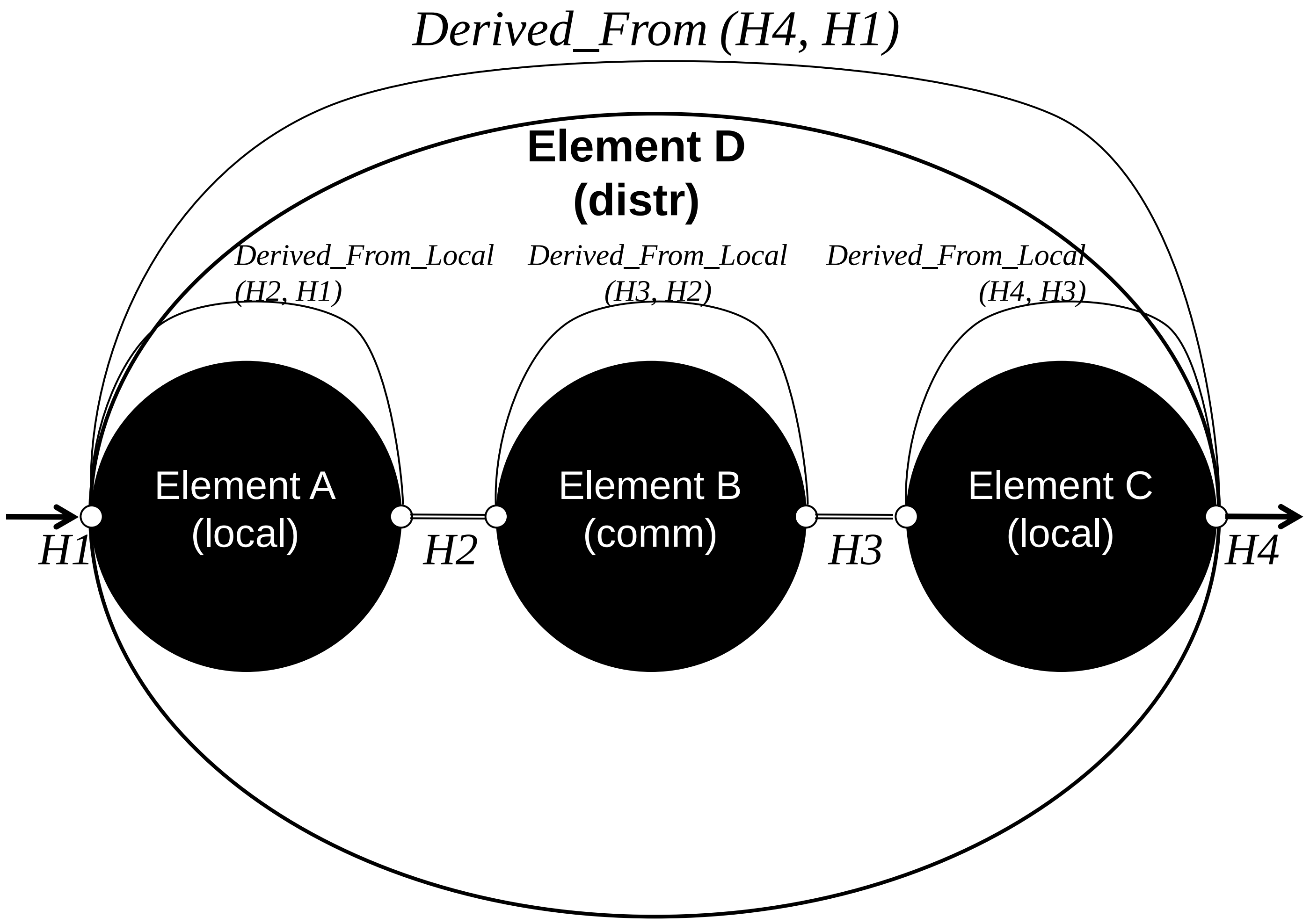
- The Pitfall
  - It's easy to end up with unsound statements.
  - Key issue: inadvertent assumption of a system-wide time referent—not a problem with "local" elements
  - Past use of "oracle functions" has defied detailed definition:
    For every entity *e2* in *H2*
    there is an entity *e1* in *H1*
    such that *e1* = *e2*
    (This ignores that *e1* may have appeared after *e2*, while we are trying to make *e1* account for *e2*!)



  - We can stay with a "mushy" definition of *Derived_From*, but that defeats any but the most minimal assurance.

Soundness of Flow History Relationships (2)

- The Plan—supporting diagram



Märzen Group

Soundness of Flow History Relationships (3)

- The Plan
  - Elaborate *Derived_From* based on a “local” element function *Derived_From_Local* .
  - This new function is applicable only to architecturally local elements, within which time is definable.
  - *Derived_From_Local* time orders all history entities appearing at its external interfaces.
  - Given the local time ordering, by which history entities have been locally time stamped, *Derived_From* associated with higher level elements (containing the related local elements) can express sound accountability relationships.
  - Remaining question: when can a communications channel be considered a local element?

Integrating BFM and State Models

- The Issue
  - Some (local) elements are best modeled using a state approach.
  - But distributed systems and subsystems need to be modeled using BFM.
  - Therefore, for a complete system security integration, the two modeling schemes must be coordinated.

Integrating BFM and State Models (2)

- The Status and Plan
  - A successful experiment:
    - Restating the GWV (state-based policy) of a separation kernel (SK) in BFM
    - Demonstrating that the BFM statement of the policy is true whenever the GWV statement of the policy is true
    - Result: The BFM scheme can validly use the claim of BFM form of the SK policy to contribute to inferring the policy of an element that contains the SK.
  - To be done
    - Perform similar experiments in other contexts (e.g., the state model of an entire platform).
    - Obtain community review of these experiments.

Tool Support for BFM

- The Need
  - Efficiency in the modeling process
  - Presentation of the model to developers, reviewers, and customers
  - Accurate validation of the model

Tool Support for BFM (2)

- The Accomplishments
  - Developed an XML-based tool.
  - The tool:
    - Accurately represents the model.
    - Allows (more-or-less) convenient capturing of model data.
    - Supports model validation.
    - Supports (marginal) graphical presentation of the model.
  - Applied the tool to a number of modeling tasks.
- Plan
  - Assess commercial tools (most are based on UML) for feasibility of add-ons to support BFM process needs.
  - Implement the add-ons and apply to modeling tasks.

CONCLUSIONS

- BFM is a feasible modeling scheme.
- BFM is workable in a number of contexts.
- BFM can be integrated with other modeling schemes.
- Claim: With adequate tool support, BFM can be used to provide necessary security assurance within production system development.