draft-xibassnez-i2nsf-capabilities-02

L. Xia, J. Strassner, C. Basile, D. Lopez

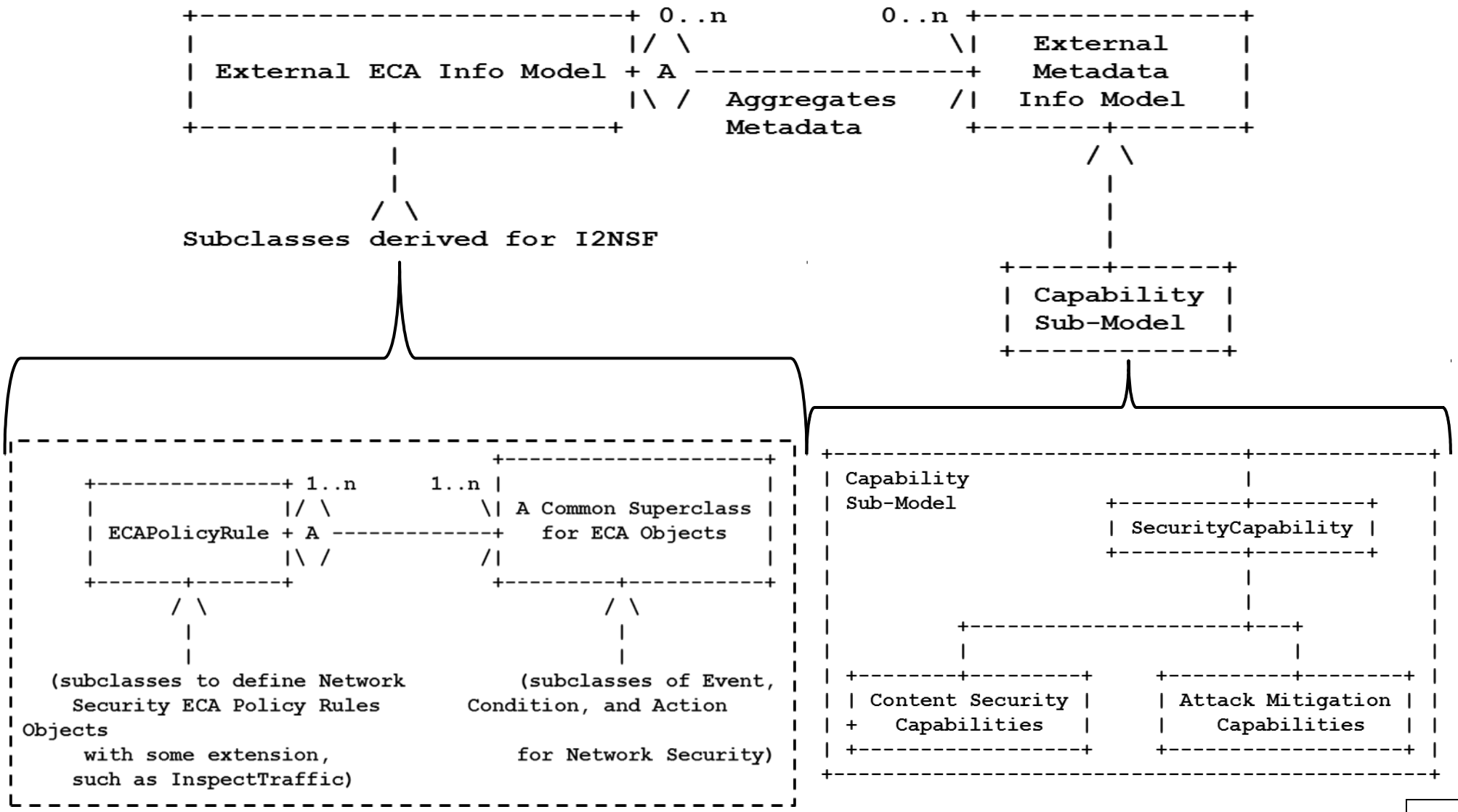
I2NSF Meeting,
Prague,
July 18th, 2017

Introduction: the Context

- **What can an NSF provide for policy enforcement?**
- **Defined by *Capabilities***
  - Capability: the functions that an NSFs provides, *independent*

    of the customer and provider interfaces
    - An abstraction with well-defined semantics
    - Flexibility to represent functionality that can be either vendor-dependent or -independent
- **This Draft**
  - Defines the concept of NSF Capabilities
    - Theory of operation and update to the Capability Algebra
  - Information model
    - Policy Rules and Capabilities

Modeling Overview



Policy Rule – Capability Duality

- **ECA Policy Rules Define Behavior**
  - *External ECA Info Model defines Rules and Rule Components*
  - *SecurityECAPolicyRule derived from External ECA Info Model*
    - Generalizes common characteristics and behavior of all I2NSF security rules
    - Subclasses refine this to provide different functionality
- **Capabilities Define Functionality**
  - *SecurityCapability subclasses from External Metadata Info Model*
    - *Defines the concept of a Capability that describes an NSF*
    - *Subclasses refine this to provide different functionality*
- **Capabilities are Manipulated by ECA Policy Rules**
  - *For example, ECA Policy Rules can define:*
    - What is or is not a Capability
    - What Capabilities can be exposed to which consumers
    - Lifecycle management of a Capability
    - Which OAM data that is exposed to which consumers

Key Point of This Draft

No need to maintain a Capability Model and a set of Policy Models for **every** NSF type.
Instead, describe the Capabilities of an NSF, and apply an appropriate policy model.

This is a scalable, model-driven approach.

The ECA Policy Rule Model

- **The Current Model Uses ECA Policy Rules**
  - *Events:* significant occurrences the NSF is able to react to
  - *Conditions:* how the NSF decides which actions to apply
  - *Actions:* actions performed by the NSF
  - *PolicyRule:* *a container that aggregates an Event, a Condition, and an Action (Boolean) clause*
- **Behavior**
  - Actions MAY execute if Event and Condition (Boolean) clauses BOTH evaluate to TRUE
  - Controlled by *resolution strategy* and *metadata*
    - Capability Algebra used to make resolution strategy decidable

Details of ECA Policy Rule Behavior

- **Policy Rules, and Policy Rule Components, are each modeled as *Reusable Objects***
- **Describe each NSF as follows:**
  - **Ac**: the set of Actions currently available from the NSF
  - **Cc**: the set of Conditions currently available from the NSF
  - **Ec**: the set of Events the NSF is able to respond to
  - **RSc**: the set of Resolution Strategies (how to resolve conflicts)
  - **Dc** defines the notion of a Default action
    - Can be a fixed action, a set of available actions, all the actions (F = full Ac), or no default action (Dc = empty set)
  - **Capability Algebra**
    - addition and subtraction of capabilities
    - ease the modelling of templates, compositions, plugins
    - asymmetric operations = union or set minus of Ac, Cc, Ec + RSc, Dc the first operand

Future Work

- **Define SecurityECAPolicyRule and SecurityCapability**
- **Appendices**
  - ***Do the Policy Rules need full object definitions before WG adoption?***
  - ***Do the Event, Condition, and Action subclasses need full object definitions before WG adoption?***
  - ***Do the Capabilities need full object definitions before WG adoption?***
- ***We need answers to questions posed in the draft***
  - ***See next 2 slides for more details***
- **Describe Exemplary Operation**
  - ***Include different examples with sample object class diagrams***

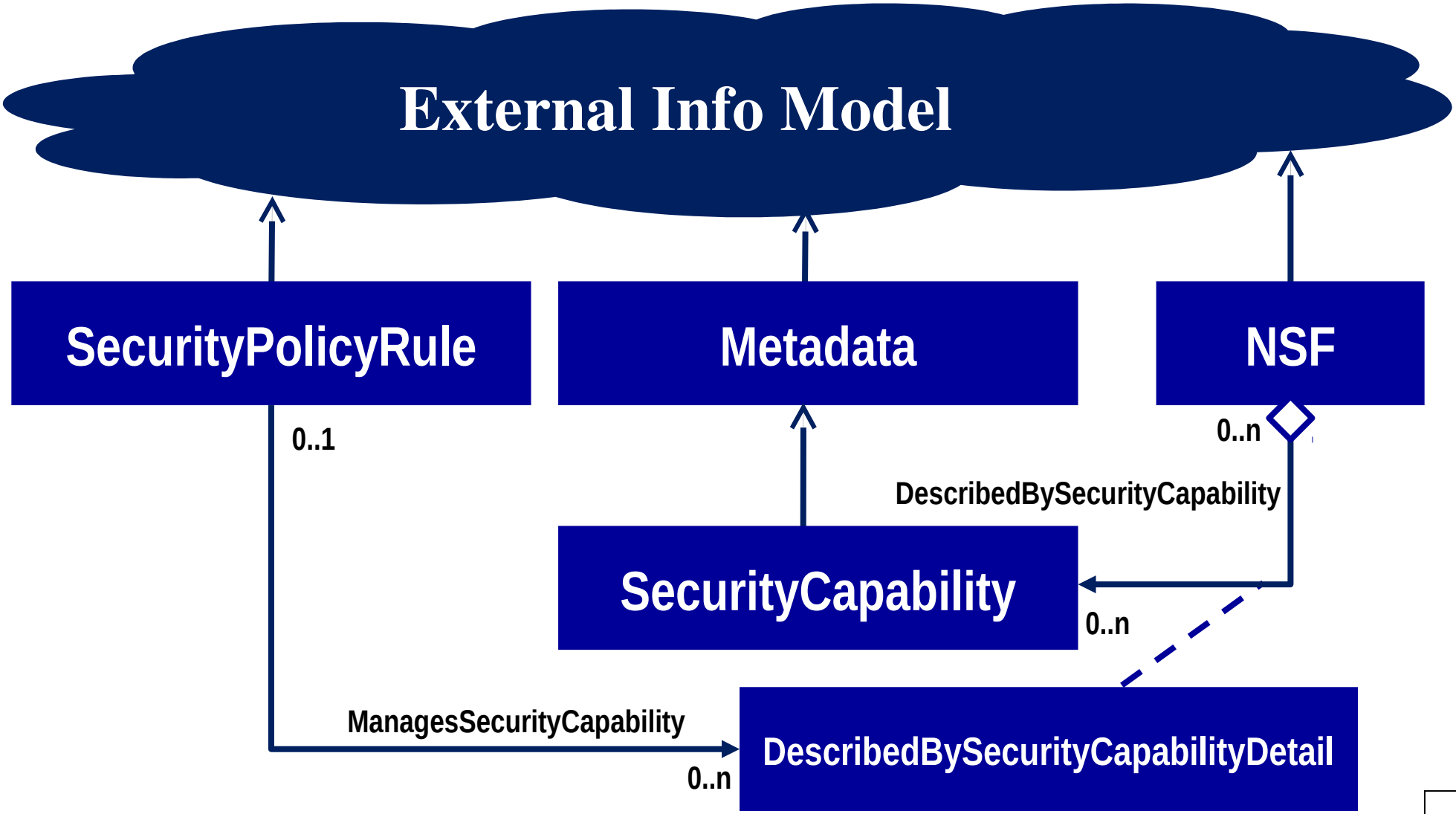
The Model: Discussion with the WG

- Possible improvements / extensions to consider for the next revision of this draft (*all questions from the I-D*)
  - Event clause / Condition clause representation
    - e.g., CNF vs. DNF for Boolean clauses
  - Event clause / Condition clause evaluation function
    - more complex expressions than simple Boolean expressions to be used
  - Action clause evaluation strategies
    - e.g., execute first action only, execute last action only, execute all actions, execute all actions until an action fails
  - More on metadata
    - authorship, time periods, (+ priorities)
  - Symmetric addition and subtraction? additional operations?

Switching to the Decorator Pattern

- **Defined categories of NSFs that need to be modelled with the Capability Model (first instantiations)**
  - based on Policy Information Models
    - Network Security Information model
    - Content Security Information model
    - Attack Mitigation Information model
- **Categories and subcategories determined with sub-classing**
  - pros: intuitive, simple, easy to design
  - cons: not very elegant, requires non-trivial maintenance at every minor update, does not work well at run-time
- **WG: should we switch to (for example) the decorator pattern?**
  - less intuitive but much more expressive, reduce classes at runtime, provides dynamic behavior (composition) instead of fragile, inheritance-based behavior (which is static)
  - More model-driven = less maintenance

Conceptual Operation (To Be Included?)



Questions?

“Create like a god. Command like a king. Work like a slave”
- Constantin Brancusi