L. Xia, J. Strassner (Huawei)
C. Basile (PoliTO)
D. Lopez (TID)

I2NSF meeting,
Chicago,
March 27th, 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 models
    - Capability, and three categories of Security Capabilities
  - Includes several discussion points for the WG
History

draft-baspez-i2nsf-capabilities-00

draft-xia-i2nsf-capability-interface-im-06

draft-xibassnez-i2nsf-capability-00

draft-xibassnez-i2nsf-capability-01

IETF96 Berlin
July 2016

IETF97 Seoul
November 2016

IETF98 Chicago
March 2017
Modeling Overview

```
+-----------------+ 0..n 0..n +-----------------+
|                 | / \                |
| External ECA Info Model + A + External | Metadata |
|                 | / \ Aggregates / | Info Model |
|                 | / | Metadata / \    |
| +-----------------+ +-----------------+  \
|                 |                   |
| Subclasses derived for I2NSF |
```

```
+-----------------+ 1..n 1..n +-----------------+
|                 | / \                |
| ECA Policy Rule + A + A Common Superclass |
|                 | / /                  |
| +-----------------+ +-----------------+  \
|                 |                   |
| (subclasses to define Network |
| Security ECA Policy Rules |
| (subclasses of Event, Condition, and Action |
| Objects with some extension, such as InspectTraffic) |
```

```
+-----------------+ +-----------------+
| Capability      | SecurityCapability |
| Sub-Model       | Attack Mitigation |
| +-----------------+ +-----------------+
| Content Security + Capabilities |
| +-----------------+ |
```

The Proposed Capability Model

- Support for ECA (and CA) Policies
  - Events
    - significant occurrences the NSF is able to react to
  - Conditions
    - how the NSF determines which actions will be applied
    - fields in packets/PDU, stateful info acquired by the NSF
    - operations available to verify condition truth (matching)
  - Actions
    - what an NSF does on packets/traffic/PDU (e.g., deny, encrypt) and related actions (e.g., logging)

- Other parameters to complete the policy specification
  - resolution strategy + external (meta)data + default action

- Templates and Capability Algebra
Details of the Proposed Capability Model

- 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 \( (F = \text{full } Ac) \), or no default action \( (Dc = \text{empty set}) \)
  - 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 of the first operand
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? Other behavior of the operations? → use cases?
**Proving Its Effectiveness**

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

This is a scalable, model-driven approach.
Questions?

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