PROMIA

Using the Principle of Least Authority to Improve Software Assurance

David Chizmadia
Promia, Inc
Senior Software Assurance Architect

410-694-0322
dchizmadia@promia.com

PROMIA

Topics

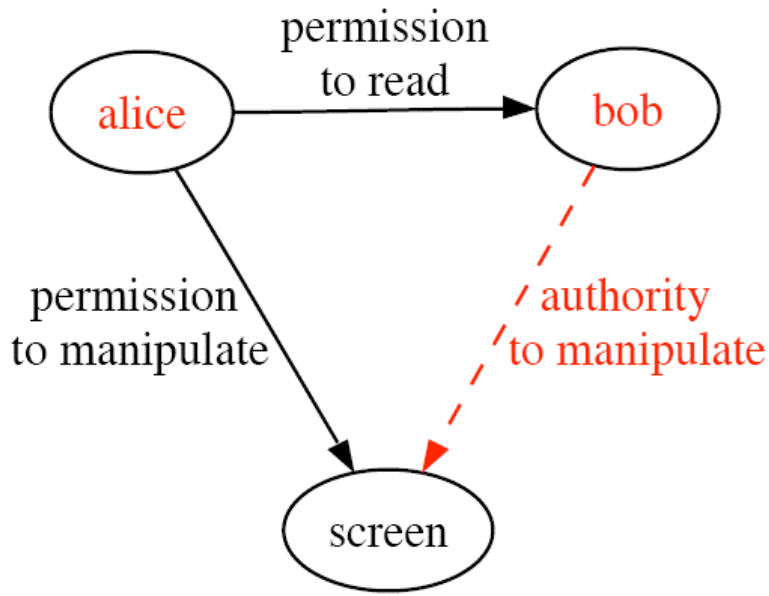
- Principle Of Least Authority (POLA) Introduction
- POLA Patterns
  - Caretaker
  - Membrane
  - Facet
  - Powerbox
- Conclusion

Brief Introduction to POLA

- **Authority versus Permission**
- ***Principle Of Least Authority (POLA)* versus *Principle Of Least Privilege/Permission (POLP)***
- **Classical (Secure) Capabilities**
- **Object Capabilities**

Authority versus Permission

- **Definitions**
  - ***Permission:*** the specific effect an entity can have in a system *as represented by the explicit ability to use another protected resource in a direct way*.
  - ***Authority:*** the general effect an entity can have in a system *as represented by both its own permissions and the permissions of other entities that will act as its proxies*.



PROMIA

POLP versus *POLA*

- **Principle Of Least Privilege**
  - **Every program and every user of the system should operate using the least set of privileges necessary to complete the job.**
- **Principle Of Least Authority**
  - **An entity should have no more authority than it needs to accomplish its specified functions**

- **Both have the same goal – limiting the damage a broken or malicious entity can cause**
  - **POLP only considers first order effects of entity**
  - **POLA considers all effects**
  - **Both are “best practice” – not immutable laws**

PROMIA

Classical (Secure) Capabilities

- **Characteristics**
  - **Protected, unforgeable, reference to an target resource or service**
  - **Possession by a caller (e.g., client) grants that caller the *capability* to interact with the target resource in specified ways**
  - **Capabilities can be obtained by:**
    - Initial conditions
    - Parenthood (for an entity that has created a new resource)
    - Reception as a parameter in a capability invocation

    Only connectivity begets connectivity
  - **System protects confidentiality, integrity, and availability of capability representation**

PROMIA

Object Capabilities

- **System in which ALL computation is performed by objects according to the capability rules**
  - **Prohibits global mutable state**
  - **Prohibits “well-known”, public services**
    - Effect of such services can be simulated by making them part of the Initial Conditions for every entity
- **Assurance Advantages**
  - **Multi-entity application is inherently modular with precisely specified interfaces between modules**
  - **Key security properties of platform can be rigorously analyzed**
  - **Information flow properties of a multi-entity application can be analyzed independently of knowledge of the code**

PROMIA

POLA
(Principle of Least Authority)
Patterns

PROMIA

Caretaker Pattern

- Major complaint about basic capability model is that it doesn't provide for access revocation
- Caretaker pattern is a variation on the Wrapper pattern
  - Caretaker holds the capability to the real object
  - Transparently (real object is not aware that Caretaker exists) forwards messages to the object
  - There exists a management capability that allows the Caretaker capability to be revoked – thus revoking access to the real object

PROMIA

Membrane Pattern

- Extension to the Caretaker pattern
- Enforces the best practice that the real object capability should not be communicated beyond the Caretaker
- Membrane wraps each capability passing through it in a subsidiary Caretaker, which is revoked with the main Caretaker

PROMIA

Facet Pattern

- Attenuates (selectively reduces) the operations available through a capability
- Facet is always a subset of the original capability
- Useful for improving robustness and safety, in addition to security
- Best practices
  - methods included in a facet should be explicitly indicated
  - if methods are added to the original interface, they should *not* be added by default to the facet

PROMIA

Powerbox Pattern

- Powerbox collects authority management into a single object
- Powerbox is arbiter of authority transfers across a complex trust boundary
- Powerbox be used for dynamic negotiation of authority during operation
- Powerbox is particularly useful when
  - the object in the less trusted realm does not always get the same authorities
  - the authorities granted to the object in the less trusted realm may change during operation

Discussion