Formalismes pour la modélisation et le test de politiques de sécurité de réseaux - MàJ

Vianney Darmaillacq
VASCO – LSR

21|03|05 – Potestat

Sommaire

Article soumis au numéro spécial sécurité du journal les *Annales des Télécommunications*

- Introduction
    - Formalismes
    - Critères de comparaison
- Exemple
- Comparaison
    - PDL
    - Ponder
    - Or-BAC
    - Tableau comparatif
- Conclusion
- Annexes : modélisations complètes

Introduction

Formalismes

- Contrôle d'accès
  - ***droits d'accès*** des ***sujets*** sur les ***objets***
  - MAC, DAC, RBAC
- Formalismes étudiés
  - PDL
  - Ponder
  - Or-BAC
- Comparaison sur un exemple

Démarche

- choix des formalismes
  - PDL
  - Ponder
  - Or-BAC
- choix des critères
  - expérience Génie Logiciel
  - Wies94, Wies95

Introduction

Critères de comparaison

- Atoms:
  - Typing, structuring
- Compositionality:
  - Consistency (conflicts), completeness (default rule)
- Expressive power:
  - Modalities, language class, reflexivity, statefulness
- Execution Model:
  - Triggering, data or goal driven
- Methodology:
  - Development phase, refinement, management scenario, lifetime, type of target, functionality of target

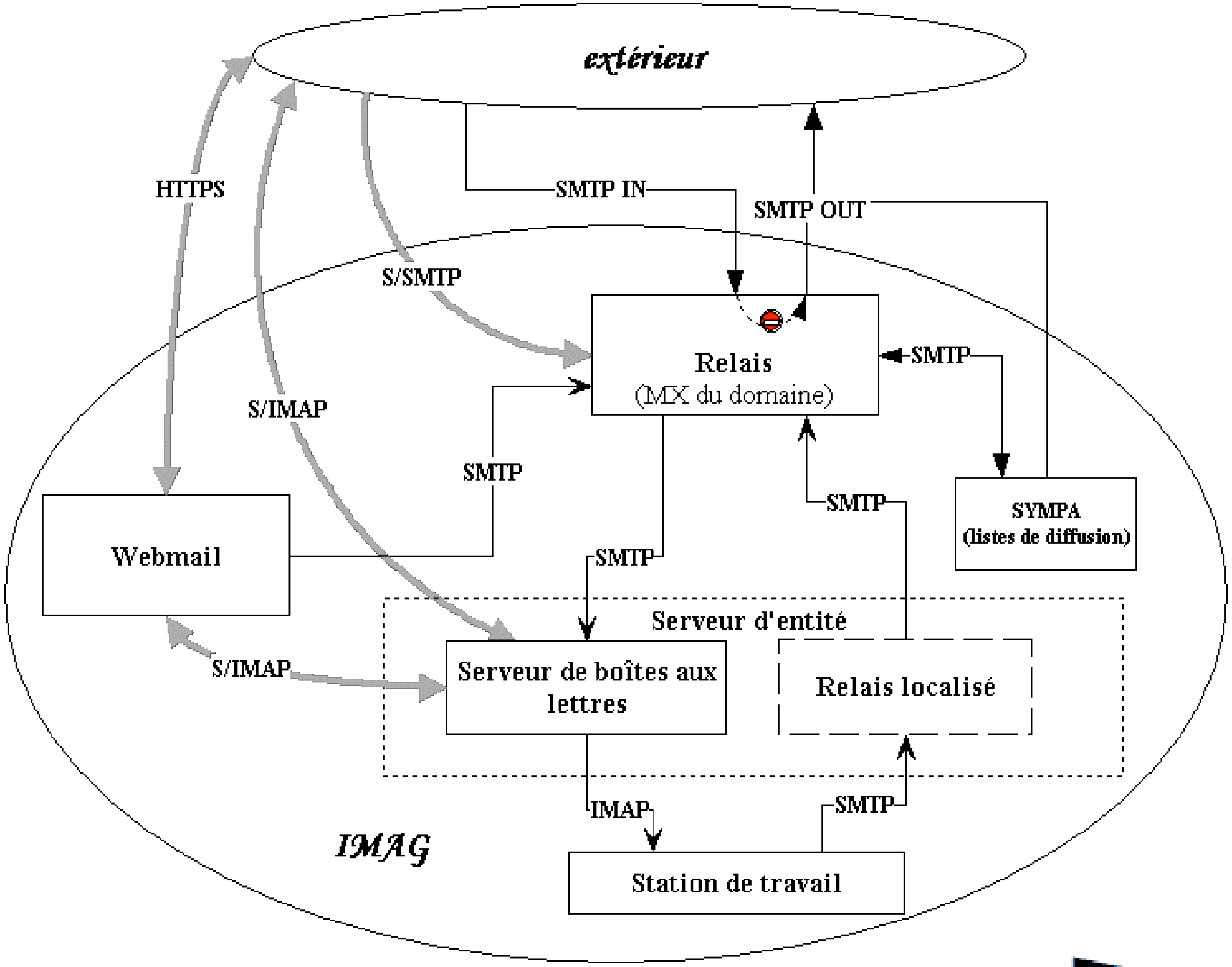
Exemple

Contexte

- Tiré de l'étude de cas IMAG
  - documents IMAG
  - règles concernant le mail
  - règles techniques
- Basé sur une découpe architecturale du réseau en zones
  - extérieur
  - intérieur
  - DMZ

Exemple

Schéma du réseau



Exemple [étude IMAG]

Politique de sécurité

1. Mail relays accepting messages from the exterior should be placed at the entry of the network, in the DMZ if possible.
2. There should be no user account on relays placed in the DMZ.
3. Mailbox servers containing user accounts should be in the private zone. There could be as many of these servers as necessary.
4. Relays in the DMZ are the only machines allowed to communicate with the exterior world using the protocol SMTP. Relay of inbound mails (to mailboxes) and of outbound mails (to exterior) is done using these relays.
5. Mailboxes could be used as internal mail relays.
6. At the entry of the site, a filtering default policy is applied, which forbid all traffic not explicitly authorized.
7. It is forbidden to relay mails from the exterior to the exterior.
8. All messages coming from the exterior are redirected to mail relays placed in entry of the site (MX field of the DNS), probably in the DMZ.

PDL

[Lobo 99]

- 2 types de règles :
  - event ***causes*** action ***if*** condition
  - policy-event ***triggers*** event ***if*** condition
- L'ensemble des *events* est une algèbre de processus. A chaque event sont attachés différents attributs (dont lieu et instant de génération, ...)
- Les *actions* sont des commandes du système et les *conditions* des fonctions booléennes. Les 2 types de fonctions prennent comme paramètres les attributs des événements.
- Exemple (étude de cas) :
  - Set of subjects S is the union of MACHINE and ACCOUNT.
  - Set of objects O is the union of MACHINE and MAIL.
  - Set of commands C = { transfer, add-account, add-mailbox }
  - Events are members of the set {request} x S x C x O
  - Actions are members of the set {grant, deny} x C.

PDL

Model

R	*PDL rule set*
1	request c=(r, transfer, r') **causes** grant(c) **if** dmz(r) **and** relay(r) **and** exterior(r') request c=(r, transfer, r') **causes** grant(c) **if** dmz(r') **and** relay(r') **and** exterior(r)
2	request c=(u, addAccount, m) **causes** deny(c) **if** dmz(m) **and** relay(m) request c=(u, connect(c), m) **causes** deny(c) **if** dmz(m) **and** relay(m)
3	request c=(u, addMailbox, n) **causes** deny(c) **if** dmz(n) request c=(r, transfer(m), r') **causes** grant(c) **if** relay(r) **and** interior(r) **and** mailbox(r') **and not** private(r') request c=(r, transfer(m), r') **causes** deny(c) **if** relay(r) **and** mailbox(r') **and not** private(r')
4	request c=(r, transfer, r') **causes** deny(c) **if not** (relay(r) **and** dmz(r)) **and** exterior(r') request c=(r, transfer, r') **causes** deny(c) **if not** (relay(r') **and** dmz(r')) **and** exterior(r)
5	request c=(r, transfer, r') **causes** grant(c) **if** mailbox(r) **and** private(r) **and** interior(r') request c=(r, transfer, r') **causes** grant(c) **if** mailbox(r') **and** private(r') **and** interior(r)
6	request c=(r, transfer, r') **causes** deny(c) **if** exterior(r) **and** interior(r') **never** grant c=(r, transfer(m), r') **and** deny(c) **monitor** (grant(c) **and** deny (c)) = grant(c)
7	request c=(m, transfer(e), m') **causes** deny(c) **if** interior(m) **and** exterior(e.src) **and** exterior(e.dst)
8	request c=(r, transfer(m), r') **triggers** redirect-mode(m) **if** exterior(r) **and** interior(r’) **and not** (relay(r’) **and** dmz(r’)) redirect-mode(m), request c=(r, transfer(m),r') **causes** grant(c)

PDL

Gestion des conflits [Chomicki 01]

- Exemple de conflit :
  - request c=(r, transfer, r') causes deny(c) if exterior(r) and interior(r')
  - request c=(r, transfer, r') causes grant(c) if dmz(r') and relay(r') and exterior(r)
- Gestion des conflits :
  - à spécifier par des contraintes :
    - ***never*** $\boldsymbol{a_1}$ ***and ... and*** $\boldsymbol{a_n}$ $\boldsymbol{a_1}$***, ... ,*** $\boldsymbol{a_n}$ ***actions***
    - exemple : C = ***never*** grant(T) ***and*** deny(T)
- Résolution des conflits :
  - Un moniteur édite les traces d'actions ou d'événements pour les rendre conformes à une contrainte :
    - M_C (grant(T) and deny(T)) = deny(T)

Ponder

[Damianou 01]

- Ponder uses the concepts of subject, target, action, condition and event.

- Authorisation/interdiction rule syntax:

```
inst ( auth+ | auth– ) policyName “{”
        subject    domain-Scope-Expression;
        target     domain-Scope-Expression;
        action     action-list;
        [ when     constraint-Expression; ] “}“
```

- Obligation rule syntax:

```
inst oblig policyName “{”
        on         event-specification;
        subject    domain-Scope-Expression;
        [ target   domain-Scope-Expression; ]
        do         obligation-action-list;
        [ catch    exception-specification; ]
        [ when     constraint-Expression; ] “}”
```

- Ponder defines the concept of domain to type objects. The set of all domains constitute a lattice organized by a partial order relationship with the semantic of membership. This relationship is noted "A/B", which means: "A/B is the set of the objects of the domain B included in the domain A" and not "the domain B is included in the domain A".

Ponder

Model

LSR
IMAG

Rqrt | *Corresponding Ponder rule set*

1 **inst auth+** P1A1 {
subject /Machine/DMZ/Relay;
action transfer(/Mail)
target /Machine/Exterior; }

inst auth+ P1A2 {
subject /Machine/Exterior;
action transfer(/Mail)
target /Machine/DMZ/Relay; }

2 **inst auth-** P2A1 {
subject /User;
action addAccount (/Account);
target /Machine/DMZ/Relay; }

inst auth- P2A2 {
subject /User;
action connect ();
target /Machine/DMZ/Relay/Account; }

3 **inst auth+** P3A1 {
subject /User;
action addMailbox (/Mailbox);
target /Machine/Private; }

inst auth+ P3A2 {
subject /Machine/Private/Mailbox;
action transfer(/Mail);
target /Machine/Private/Station; }

inst auth+ P3A3 {
subject /Machine/Interior/Relay;
action transfer(/Mail);
target /Machine/Private/Mailbox; }

4 **inst auth-** P4A1 {
subject /Machine/Interior;
action transfer(/Mail);
target /Machine/Exterior; }

inst auth- P4A2 {
subject /Machine/Exterior;
action transfer(/Mail);
target /Machine/Interior; }

5 **inst auth+** P5A1 {
subject /Machine/Private/Mailbox;
action transfer(/Mail);
target /Machine/Interior;}

inst auth+ P5A2 {
subject /Machine/Interior;
action transfer(/Mail);
target /Machine/Private/Mailbox;}

6 **inst auth-** P6A {
subject /Machine/Exterior;
action transfer (/Mail);
target /Machine/Interior; }

inst meta P6M **raises** R.action {
exists(R I R.type=/Rule **and** R.subject==/Machine/Exterior
and R.action==transfer(/Mail) **and** R.target==/Machine/Interior);
R.modality == '"auth+"; }

7|8 **inst auth-** P7A {
subject /Machine/Interior;
action transfer (m=/Mail);
target /Machine/Exterior;
when m.src.isExt() **and** m.dest.isExt(); }

inst oblig P8O {
on R.transfer(m=/Mail) to R'=/Machine/Interior);
subject /Machine;
do transfer(m) to /Machine/DMZ/Relay;
when R == /Machine/Exterior **and** R' != /Machine/DMZ/Relay; }

Ponder

Gestion des conflits

- Règle par défaut :
  -
```
inst auth- P6A {
  subject  /Machine/Exterior;
  action    transfer (/Mail);
  target   /Machine/Interior; }
```

- Relation de précédence sur les règles selon les modalités :
  -
```
inst meta P6M raises R.action {
  exists(R | R.type=/Rule and
     R.subject ==   /Machine/Exterior and
     R.action    ==   transfer(/Mail) and
     R.target    ==  /Machine/Interior );
  R.modality == "auth+"; }
```

Or-BAC

Concepts [Cuppens 03]

- Subject --> Role
  Action --> Activity
  Object --> View
- Étude de cas :
  - subjects = objects = { machine }
  - actions = {transfer}
  - roles = {relay, mailbox, station, any-machine}
  - activities = {any-activity, relaying, mail-boxing, working, ext-relaying}
  - views = {any-machine, workstation, mailbox, relay, int-machine, ext-machine}

Or-BAC

Modélisation

Rqrt	*Or-BAC rule set*
1	permission (dmz, relay, relaying, any-machine)
2	prohibition (dmz, relay, mail-boxing, any-machine)
3	permission (private, mailbox, mail-boxing, workstation) permission (private, mailbox, mail-boxing, relay) prohibition (dmz, int-machine, mail-boxing, any-machine)
4	prohibition (interior, int-machine, relaying, ext-machine, not relay-in-dmz)
5	permission (interior, mailbox, relaying, int-machine)
6	prohibition (dmz, int-machine, any-activity, any-machine)
7	prohibition (interior, int-machine, ext-relaying, any-machine)
8	obligation (interior, any-machine, redirecting, relay, not-relay-and-receive)

Or-BAC

Résolution des conflits

- Détection syntaxique des conflits
  - Rôles/contextes différents
  - Héritage des règles associées à un rôle
  - Nouvelle règle insérée dans la base
- Résolution des conflits
  - Transformation en logique du premier ordre
  - Base de règles priorisée
  - Obtention d'ensemble maximaux de règles non conflictuelles applicables

Analyse

Table comparative

Criteria	*PDL*	*Ponder*	*Or-BAC*
Typing	condition part	domain typing	by abstracting and contexts
Structuring	process algebra	lattice on domains tree inheritance on roles	lattice on organizations, roles, views, activities
Default rule (completeness)	no	should be specified	restrictive policy by default
Inter-rules conflicts (consistence)	manual specification of conflicts manual resolution of conflicts	syntactic detection of conflicts precedence relation to resolve conflicts	syntactic detection of conflicts precedence relation to resolve conflicts
Modalities	triggered obligation	permission, interdiction triggered obligation refrain, delegation	permission, interdiction obligation
Reflexivity	no	yes	administration model
Statefulness	yes	no	no
Triggering	easy apart in case of use of pseudo-permission	requires a mapping	mapping using abstractions
Data/goal driven	data driven	data driven	data driven

Conclusion

Bilan général

- Chaque formalisme permet de représenter les besoins informels de la politique

- Les formalismes existants insistent sur le caractère déclaratif et incohérent d'une politique de sécurité

- Pas d'exemples de :
  - politique de niveau intermédiaire
  - raffinement d'une politique de haut niveau en une politique de bas niveau
  - mise en conformité de matériels ou logiciels à une politique de sécurité de haut niveau

Conclusion

Perspectives

- étendre l'étude de cas
  - besoins moins techniques
- tester les systèmes par rapport à une politique formelle
  - identification des niveaux de détail pertinents
  - choix des formalismes
  - relations de conformité