Authorization Using the Publish-Subscribe Model

Qiang Wei, Matei Ripeanu, Konstantin Beznosov

Laboratory for Education and Research in Secure Systems Engineering
lersse.ece.ubc.ca
Department of Electrical and Computer Engineering
outline

- the overview
- system design
- evaluation
- summary & future work
authorization (access control) architecture

This architecture is based on the **point-to-point model**, used by IBM Access Manager, Entrust GetAccess, CA SiteMinder, etc.

- **Subject** (client) sends an **application request** to the **application server**.
- The **application server** sends an **application response**.
- The **application server** sends an **authorization request** to the **policy enforcement point (PEP)**.
- The **PEP** makes a **policy decision** (enforcement point, PDP) and sends an **authorization response**.
- The **PDP** sends a **(request, allow) authorization** response to the **application server**.
- The **application server** sends a **policy decision point (PDP)** response to the **PEP**.
- The **PEP** sends an **application response**.
- The **subject** (client) receives the **application response**.
problem motivation

+ consistent policy enforcement
+ re-use of authorization logic

- reduced availability
- reduced scalability
- high management overhead
in large-scale systems

fragile
difficult to maintain
addressing the problem: big picture

authorization requests ↔ authorization responses

1. publish-subscribe
2. active recycling
3. speculative precomputing
the contribution of this paper

- study the use of a publish/subscribe (pub-sub) channel between PEPs and PDPs
  - design system architecture and data flow
  - analyze the expected benefits
  - propose the pub-sub requirements and the methods to meet these requirements
- evaluate system availability improvement and performance
basic components in a publish/subscribe system

- publishers
  - producer 1
  - producer 2
  - producer 3

- event

- notification service (ENS)

- subscribers
  - consumer A
  - consumer B

event publishers

notification

subscribe

subscribe
related work

publish-subscribe applications

- Internet games
- WWW update
- software monitoring
- access control

publish-subscribe systems

- Scribe
- Gryphon
- Elvin
- Siena
- ...
outline

- the overview
- system design
- evaluation
- summary & future work
system architecture

event notification service (ENS)

subscribe to request, publish response

subscribe to response, publish request

PEP

PDP

PDP

PEP

PEP

response notification
8. notify the response

3. subscribe to the response for the request

4. publish the request

5. notify the request

7. publish the response

1. subscribe to all the requests it can resolve

6. compute the response

2. generate a request

9. enforce the response

10. unsubscribe to the response

PEP

PDP

event notification service (ENS)
expected benefits

- increased availability
- reduced administration overhead
- reduced integration costs

PEP  
response  
ENS  
request  
PDP
ENS requirements

- robustness
  - distributed ENS
- security
  - integrity
- performance
  - optimization techniques
performance optimization I: alternative data flow

1. subscribe to all the requests it can resolve
2. generates a request
3. subscribe to the response for the request
4. publish the request
5. notify the request
6. compute the response
7. publish the response
8. notify the response
9. enforce the response
10. unsubscribe to the response
performance optimization II: using approximate recycling

SDP: secondary decision point

The number of calls to ENS is reduced

Event notification service (ENS)
outline

- the overview
- system design
- evaluation
- summary & future work
use Siena as the ENS

- popular publish-subscribe system
- designed for wide-area networks
- implemented, available and maintained
evaluating availability improvement

- setup
  - multiple PDPs with resource overlap
  - each PDP failed after some time and required some time to repair
  - time-to-failure (TTF) and time-to-repair (TTR) followed an exponential distribution

- metric
  - percentage decrease of failed requests =

\[
\frac{|\text{failed\_requests}|_{\text{point-to-point}} - |\text{failed\_requests}|_{\text{pub-sub}}}{|\text{failed\_requests}|_{\text{point-to-point}}}
\]
evaluation results

the impact of the number of PDPs

10% overlap rate between PDP resource

the impact of the overlap rate

4 PDPs
evaluating performance impact

- metric
  - response time
    - the time between the event that the PEP sends a request and the event that the PEP receives the response

- questions
  - how does our design perform?
  - how do the proposed performance optimization techniques help?
response time comparison

- point-to-point
- pub-sub with alternative data flow
- pub-sub

average response time (ms)
reduced response time by adding SDPs

PDP is local and fast

PDP is remote or slow

pub-sub without SDP

adding SDP
outline

- the overview
- system design
- evaluation
- summary & future work
summary

fragile
difficult of manage

increases with the number of PDPs

pub-sub without SDP

adding SDP
future work

- large scale experiments
  - a distributed ENS
  - multiple PEPs and PDPs
- comprehensive security mechanisms
  - threat model
  - defense techniques