

Fine-grained Security Management in a Service-oriented Grid Architecture

S. Mueller ^(1,2), A. Hoheisel ⁽¹⁾ and B. Schnor ⁽²⁾

(1) Fraunhofer Institute for Computer Architecture and Software Technology (FIRST), Berlin, Germany

(2) Institute for Computer Science, University of Potsdam, Germany

16. October 2006

Outline

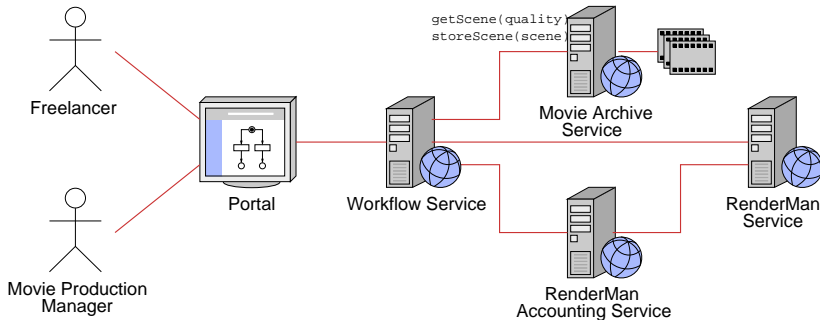
Motivation

Security Architecture

Comparison

Conclusion

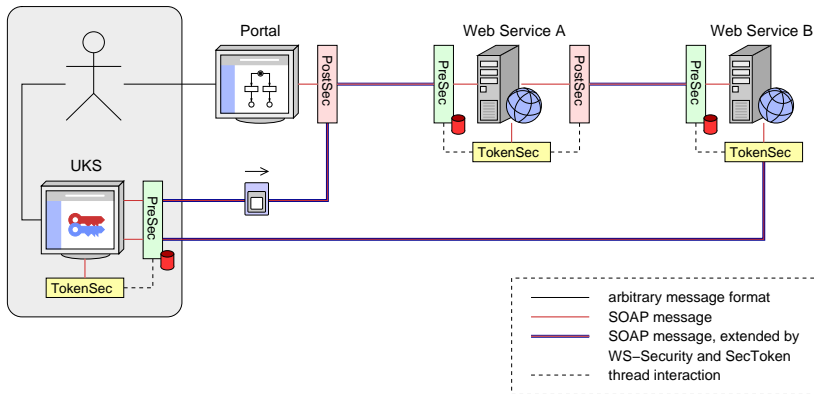
Media Industry



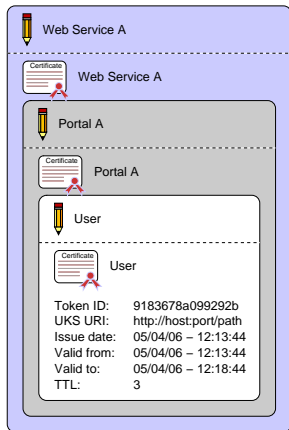
Requirements

- ▶ access control
 - ▶ Role Based Access Control (RBAC)
 - ▶ **trace of intermediate stations** incorporates into the authorisation decision
 - ▶ fine-grained to the point of **SOAP messages and their parameters**
- ▶ restricted delegation
 - ▶ **user maintains control of his credentials**
- ▶ convenient integration
 - ▶ independent of SOAP implementation
 - ▶ security out of the box

Overview



Security Token

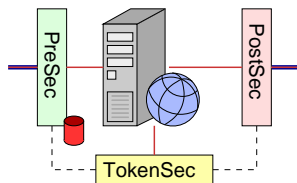


```

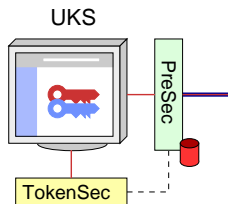
- <TokenSignature>
+ <Signature/>
+ <Certificate/>
- <TokenSignature>
+ <Signature/>
+ <Certificate/>
- <TokenSignature>
+ <Signature/>
+ <Certificate/>
  <TokenData tokenID="..." dvsURI="..."
    issueDate="..." validFrom="..."
    validUntil="..." ttl="...">
  </TokenSignature>
</TokenSignature>
</TokenSignature>
  
```

Security Components

- ▶ PreSec
 - ▶ authentication and decryption
 - ▶ extract and verify SecToken
 - ▶ perform authorisation
- ▶ TokenSec
 - ▶ sign the SecToken
 - ▶ SOAP interface for user credentials and message context
- ▶ PostSec
 - ▶ attach SecToken
 - ▶ sign and encrypt the message



User Keystore Service (UKS)



- ▶ manage certificates, private keys and other credentials
- ▶ SOAP interface for obtaining SecToken and Credentials
- ▶ protected by the same security mechanisms as common services are protected by
- ▶ credential owner defines access rules

Policy Language

- ▶ eXtensible Access Control Markup Language (XACML)
- ▶ Request/Response Language
- ▶ Policy Language divided into PolicySets, Policies and Rules
- ▶ reference implementation: SunXACML
 - ▶ API: construction of requests and policies
 - ▶ ready to use Policy Decision Point (PDP)

How We Do Compare

Category	Property	GSI	Unicore	YAGSI ⁽¹⁾
authentication	SSL/TLS	✓	✓	✗
	WS-Security	✓	✗	✓
authorisation	ACL	✓	✗	✗
	RBAC	✗	✓	✓
delegation	supported	✓	✗	✓
	under control of user	✗	✗	✓

(1) Yet Another Grid Security Infrastructure

Planned Application

We plan to apply this security infrastructure in several international and national projects such as:

- ▶ CoreGRID
- ▶ K-Wf Grid
- ▶ MediGRID
- ▶ Instant-Grid

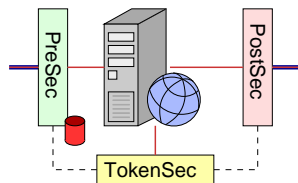
Summary

- ▶ conventional Grid security architectures focus on the service provider's perspective and do not provide fine-grained authorisation
- ▶ our approach overcomes this drawbacks
 - ▶ credentials stay under full control of the owner
 - ▶ security token discloses user and intermediate stations
 - ▶ security out of the box
- ▶ state of implementation
 - ▶ prototype we be available in march 2007

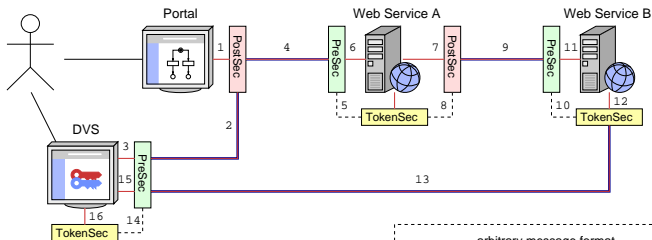
Thank you for your attention!

Security Components





- ▶ PreSec
 - ▶ authentication and decryption
 - ▶ extract and verify SecToken
 - ▶ perform authorisation
- ▶ TokenSec
 - ▶ sign the SecToken
 - ▶ SOAP interface for user credentials and message context
- ▶ PostSec
 - ▶ attach SecToken
 - ▶ sign and encrypt the message



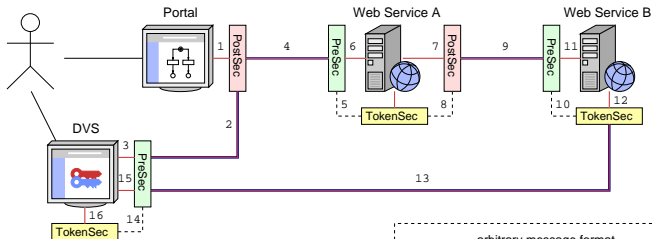
Applying the Security Infrastructure





1. `a.foo(..., dvsURI, userID, pass)`
2. `dvs.genSecToken(userID, pass)`
3. `genSecToken(userDN)`
4. `foo(..., tokenID) + `
5. `storeToken(tokenID, token)`
6. `foo(..., tokenID)`
7. `b.bar(..., tokenID)`
8. `getToken(tokenID)`

 arbitrary message format
 SOAP message
 SOAP message, extended by WS-Security and SecToken
 thread interaction

Applying the Security Infrastructure



9. `bar(..., tokenID) + `
10. `storeToken(tokenID, token)`
11. `bar(..., tokenID)`
12. `secToken.getCredential(tokenID, credentialID)`
13. `getCredential(tokenID, credentialID) + `
14. `storeToken(tokenID, token)`
15. `getCredential(tokenID, credentialID)`
16. `getUserDN(tokenID)`

