

Konzepte einer Sicherheitsarchitektur für eine SOA am Beispiel der eFA

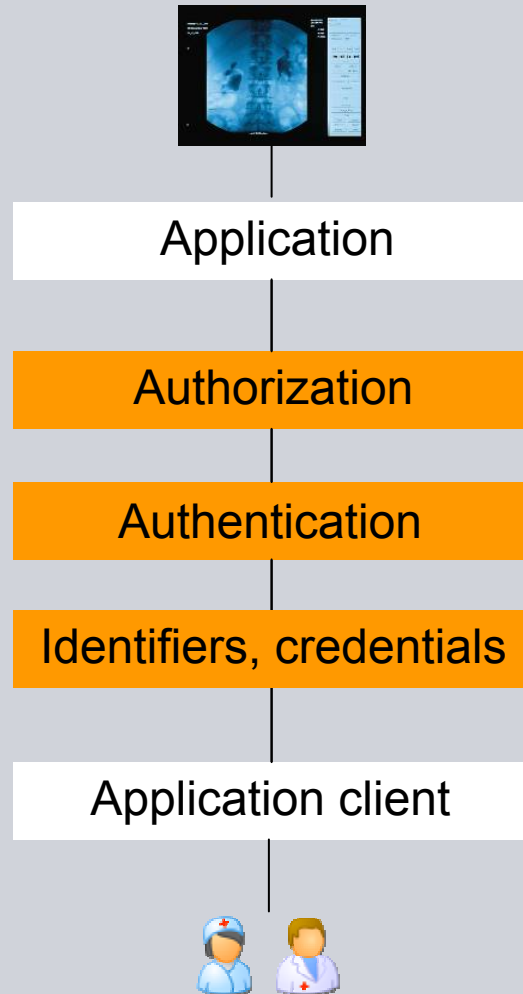
SOA Security - So What?

BITKOM Workshop SOA&Security, Frankfurt/Main 2008-03-12

Contents

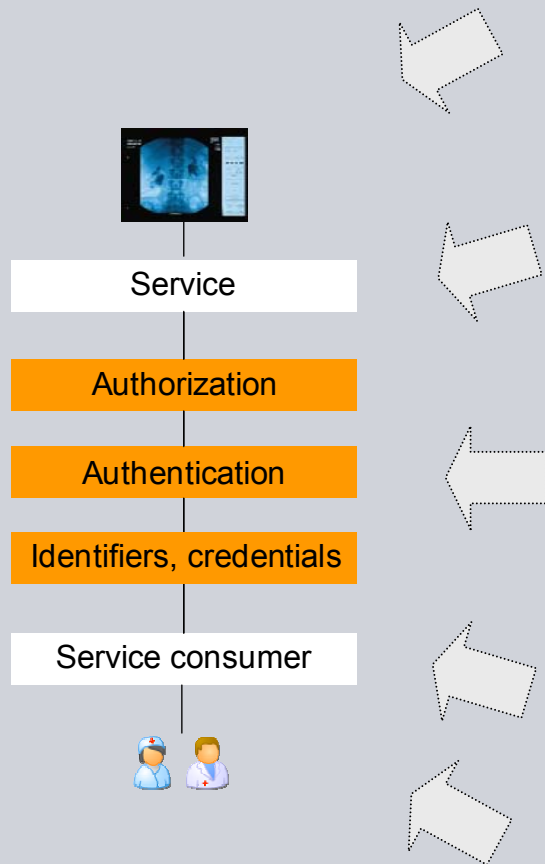
- Setting the Scene
- Architectural Recipe
- Solution Blueprint
- WS-Stack Integration
- Spotting eFA on the Radar-Screen / About the Siemens Realization for eFA Security
- Conclusions

The IT-Security Issue Before SOA



- A “law of nature” in IT:
 - Serving valuable resources in an automated fashion mandates authorization
 - Authorization calls for authentication
 - Authentication depends on identifiers, credentials
- This applies to every IT-system like gravity applies to any spot on Earth. It especially holds for:
 - Health care e.g. eFA
 - ...
 - SOA

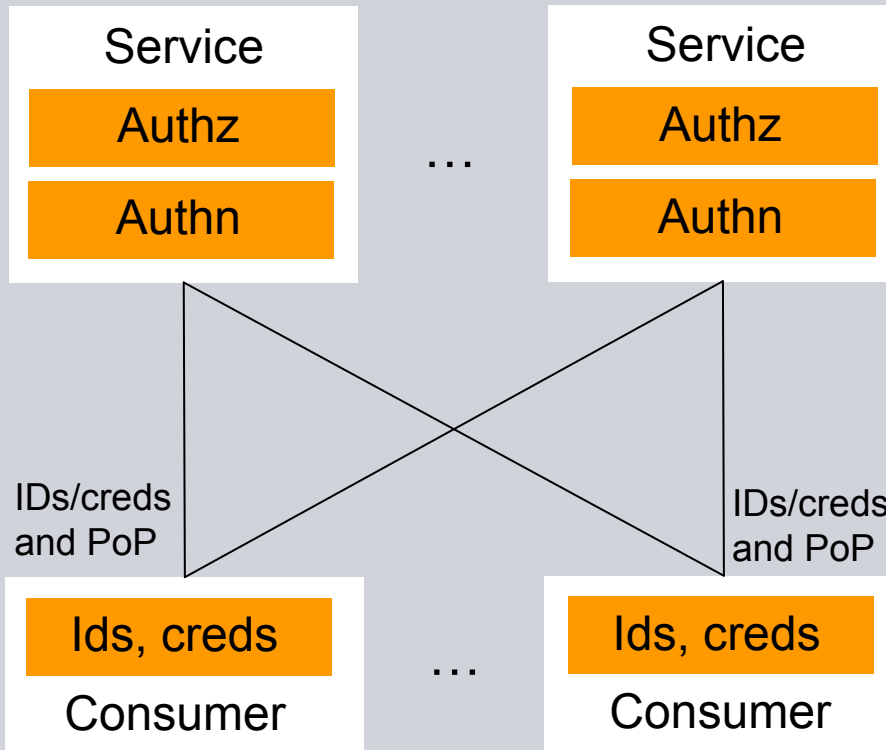
How Does SOA Change the Picture?



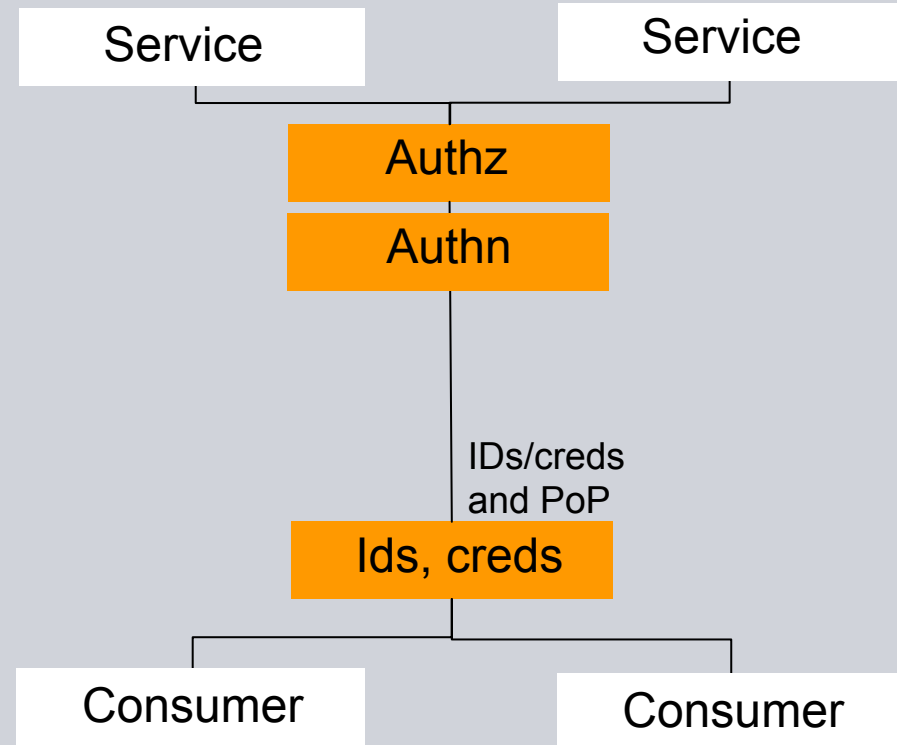
- Dynamics:
 - Require opening-up user populations that can be served (identity federation) and related authorization models (e.g. attribute-based authorization).
- Loose coupling:
 - Traditional security architectures with their rigid coupling between authorization (consumer of authenticated subject information) and authentication (producer of such information) are incompatible with SOA.
- Heterogeneity:
 - Security aspects are being modeled as part of the service contract. This is based on a declarative modeling of cryptographic protocols and required security tokens.
- Innovation:
 - WS-based SOA need to adapt XML security technologies e.g. WS-Security, WS-Trust, SAML, XACML.
- Further influences: Web 2.0 concepts such as user-centric identity (CardSpace, OpenID) and self-determination (also relevant in health care: my body→my data→my control).

Externalize Security as a Cross-Cutting Concern

Naïve approach: *DIY*



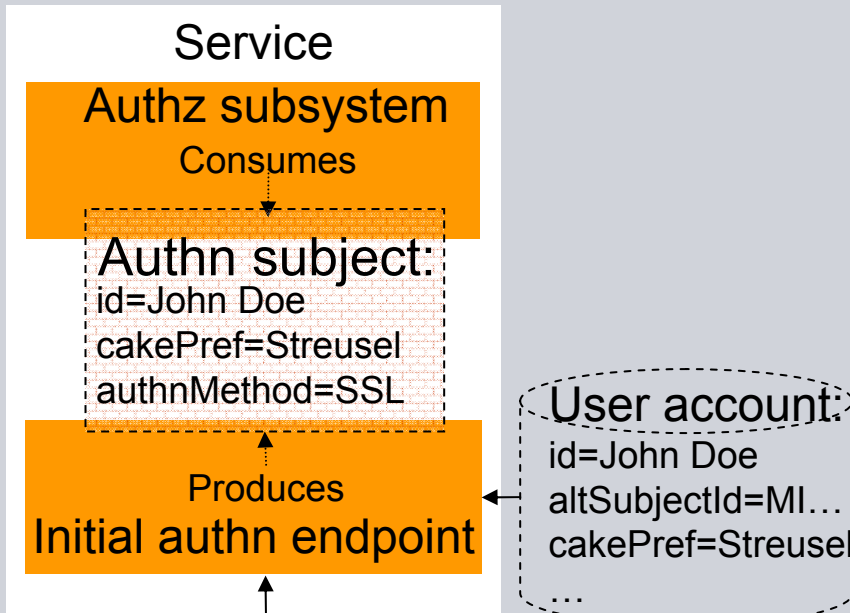
Advanced approach: *re-use*



- ☹️ Coupling between supply and use of security
- ☹️ Duplication of work
- ☹️ ...

Decouple Authorization from Initial Authentication

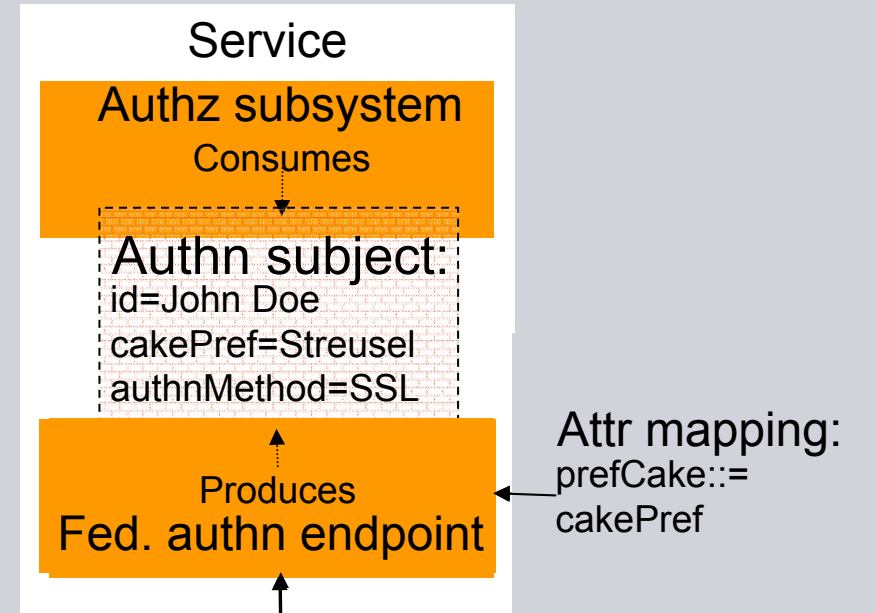
Traditional approach: *piggybacked*



Initial authn protocol:
 Cert=MI...
 PoP=SSLSign(SrvNonce)

- ☹ Causes identity enclaves
- ☹ Mandates RPs to be IdPs
- ☹ ...

Federated approach: *split work*

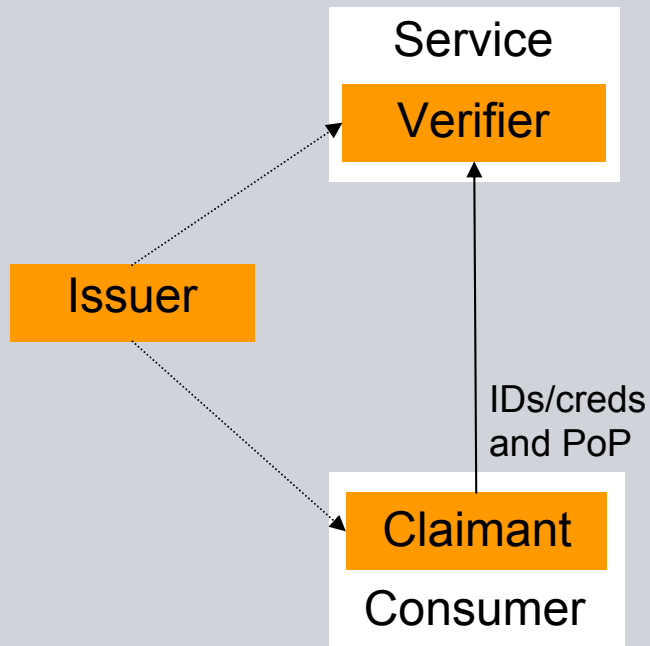


Federated authn protocol:
 Assertion=<id=John Doe, prefCake=Streusel>
 PoP=WSSESign(SrvNonce)

Initial authn endpoint

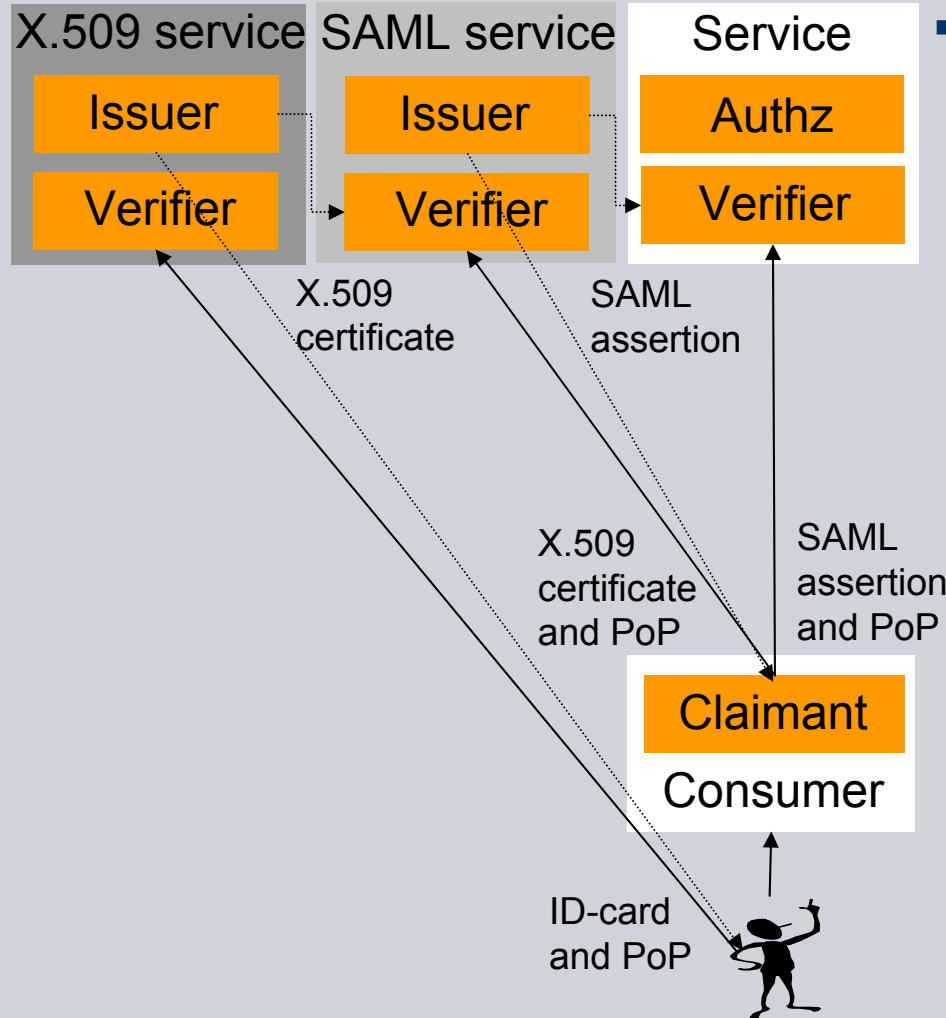
Initial authn protocol

Authentication Subsystem - What to Consider?



- Considered authentication subsystem artifacts:
 - Issuer: issues reference information (e.g. identifiers, credentials)
 - Claimant: claims identity and needs to prove it
 - Verifier: verifies identity claims and their proofs
- Work-split:
 - Issuer: (typically) external
 - Claimant: service consumer
 - Verifier: needs to reside in the service application or its call stack

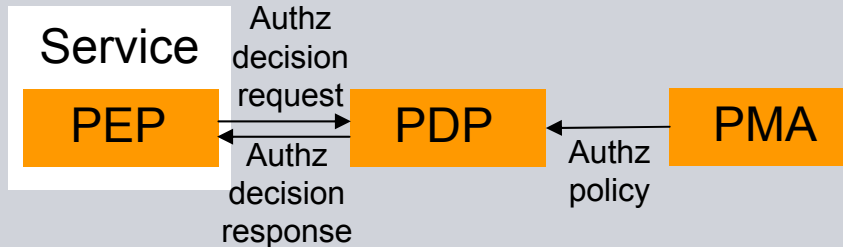
Authentication Subsystem - How to Employ?



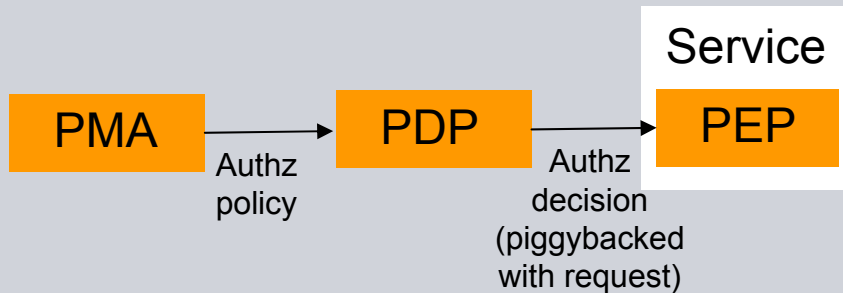
- In practice, this basic pattern is iterated e.g.:
 - Tertiary authn: via SAML assertions (prerequisite for service access)
 - Secondary authn: via X.509 certificates (prerequisite for SAML assertion issuance)
 - Primary authn: via ID-cards (prerequisite for X.509 certificate issuance)
- Architectural trick:
 - Treat internal user populations with the same approach by e.g. introducing a password-based authentication service issuing SAML assertions.
 - Simplifies verifiers by requiring them to support either initial authn (with local user accounts) or federated authentication.

Authorization Subsystem - What to Consider?

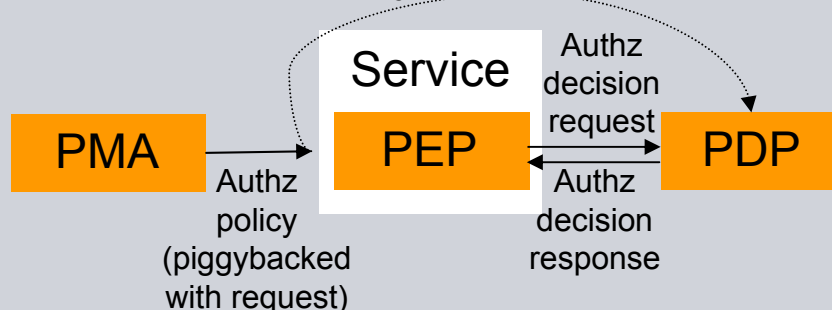
Authorization decision pull:



Authorization decision push:



Authorization policy push:



■ Considered authorization subsystem artifacts:

- PEP: Policy Enforcement Point
- PDP: Policy Decision Point
- PMA: Policy Management Authority

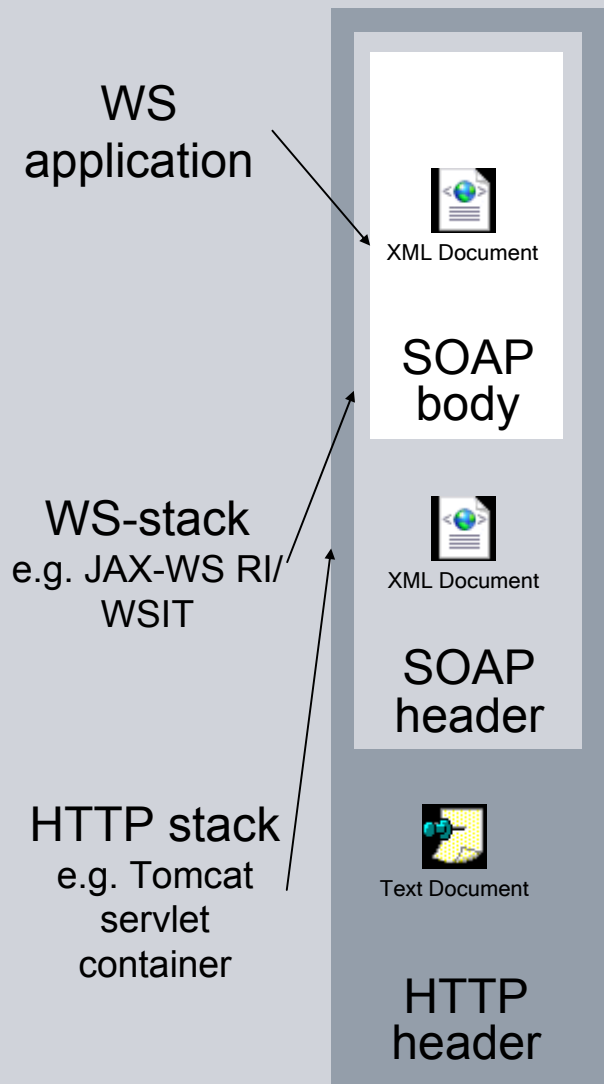
■ Work-split:

- PEP: needs to reside in the service application or its call stack
- PDP: can be externalized from the service application and its call stack (via local or remote communications)
- PMA: can be externalized from the service application and its call stack

■ These options apply to various models:

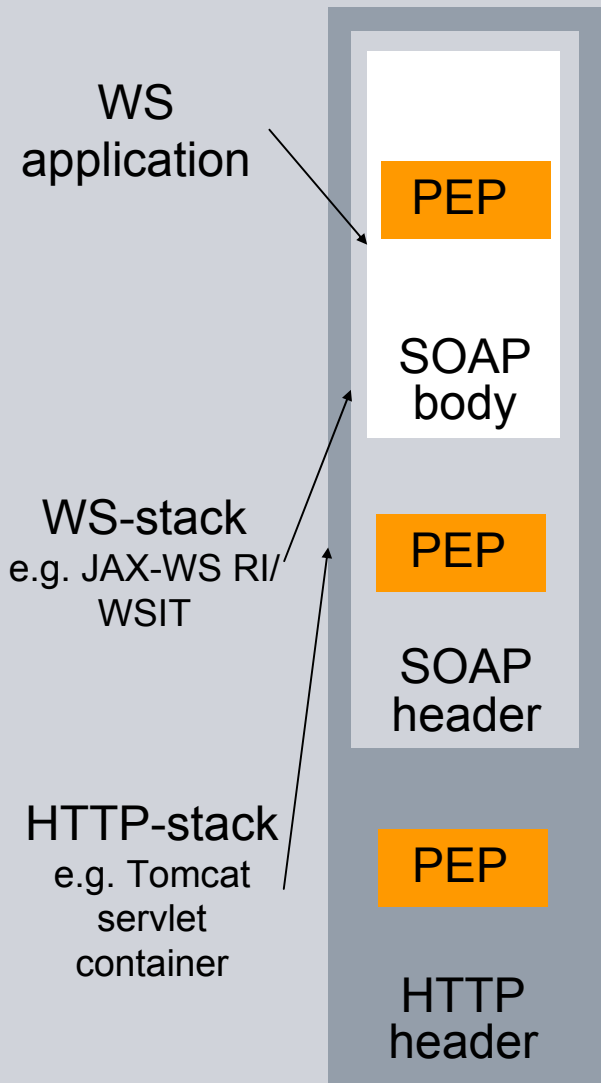
- Authorization decision pull
- Authorization decision push
- Authorization policy push

Authorization Subsystem – Which Expressiveness?



- Subject-related abstractions:
 - Authentication subsystem-specific
 - May comprise information managed in external domains.
 - In eFA: HL7 II elements with organization and role affiliations of a subject
 - Resource-related abstractions:
 - Application-specific resource identifiers and attributes.
 - WS addressing information
 - In eFA: HL7 II elements specifying information objects
 - Action-related abstractions:
 - Application-specific action identifiers and attributes
 - WS operation information
 - In eFA: names of eFA WS primitives
- This makes the case for attribute-based authorization:
- Subject identifiers and attributes managed in external domains need to be handled to cover identity federation
 - An application-specific vocabulary needs to be handled to deliver application-aware authorization

Authorization Subsystem - How to Employ?



- PEPs residing in WS applications:
 - ☺ In case of AOP, no impact on application sources; need to integrate API calls otherwise
 - ☺ Allows arbitrarily fine-grained request authorization (may subject X call method Y upon resource Z?)
 - ☹ Needs to consume authentication state from WS-stack
- PEPs residing in WS-stacks:
 - ☺ Decoupled from application sources or binaries
 - ☹ Allows fine-grained request authorization as far as authorization-relevant information is available in request
 - ☺ Supports WS authentication protocols
 - ☹ Relies on WS-stack to hand-over properties to application
- PEPs residing in HTTP-stacks:
 - ☺ Decoupled from application sources or binaries
 - ☹ Limited to coarse-grained request authorization (may subject X access the service application?)
 - ☹ Limited to HTTP authentication protocols
 - ☹ Hard to hand-over properties to application



WS-Stack Integration

SIEMENS

- WS-based SOA systems typically build upon off-the-shelf WS-stacks e.g. Apache Axis, Microsoft WCF, Sun JAX-WS RI/WSIT.
- Authorization properties of these WS-stacks:
 - All support externalization through an extensible SOAP handler-chain pipeline. This allows to employ an authorization subsystem via PEPs.
 - Microsoft WCF provides native support for claim-centric authorization. If this does not meet given authorization requirements, externalization can be realized on several levels (PEP, PDP and/or PMA).
- Authentication properties of these WS-stacks:
 - All support the processing of WSSE-defined identifiers and credentials as well as their PoP.
 - All allow augmenting an authenticated subject identity in an application-specific way through an extensible SOAP handler-chain pipeline.
 - All allow propagating authenticated identity towards service applications.

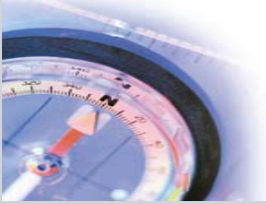


Spotting eFA on the Radar-Screen

Aspect	eFA specification
Externalizing security as a cross-cutting concern	Separates medical application architecture from security architecture
Decoupling authorization from initial authentication	Allows to isolate endpoints for verifying initial authentication based on X.509 certificates Requires application services to "only" process SAML assertions issued by eFA
Authorization architecture	Relies on a DAC authorization model addressing patient consent (my body→my data→my control) Modeled according authorization policy push PEPs may reside in WS-stacks or the service applications (e.g. through AOP) Requires a fine-grained SOAP request parsing to lookup identifiers and match them
Authentication architecture	Relies on an n-ary authentication architecture where: - eFA application services: consume SAML assertions plus PoP - eFA security services: issue SAML assertions and consume X.509 certificates plus PoP - Ext. security services: issue X.509 certificates and consume whatever is appropriate given their CPS Note that this simplifies things somewhat as eFA security is based on multiple SAML assertions (cf. below) and adds authentication architecture artifacts issuing SAML assertions while consuming (other) SAML assertions plus PoP
Adaptation to technology innovation	Relies on SAML, SOAP Message Security, WS-SecurityPolicy, WS-Trust, XACML Does not yet use WSFED
Work split between architectural artifacts	eFA IdentityProvider STS: encapsulates the processing of X.509 certificates and access to persisted user data eFA ECRAccessTokenService: encapsulates the pseudonymization of a patient and health professional context eFA ECRAdmissionTokenService: encapsulates the look-up of authorization policies
Separation of functional concerns	Distinguishes between: - Health pro-determined operations: eFA IdentityProvider STS - Health pro and patient-determined operations: eFA ECRAccessTokenService - Health pro, patient and ECR-determined operations: eFA ECRAdmissionTokenService Note: handling of multiple SAML assertions (in one ECR/MDO request context) is an implication of this separation
Potential of re-use	eFA IdentityProvider STS: WS-Trust STS with specific WSDL and SAML assertion profile eFA ECRAccessTokenService: eFA-specific business logic eFA ECRAdmissionTokenService: nucleus for an authorization policy push support in WS environments eFA "PEPs": somewhat eFA-specific since they need to understand eFA application service primitives (to some degree) and the eFA SAML assertion vocabulary

About the Siemens Realization for eFA Security

- Functional properties:
 - Java 1.6-based realization of the eFA security subsystem
 - JAX-WS RI/WSIT is being used for SOAP header processing
 - Actual request processing done in own code (incl. own WS-Trust STSs)
 - eFA IdentityProvider STS, ECRAdmissionTokenService, ECRAccessTokenService offload processing tasks to a security server
 - eFA “PEPs” reside as implementations of the JAX-WS `SOAPHandler` interface in the WS-stack. They use the JAX-WS `SOAPMessageContext` to transfer security-related information to the actual applications.
- Non-functional properties (here: approx. added security overhead):
 - eFA identity / admission / access assertion acquisition: 0,18 / 0,11 / 0,15 s
 - eFA application service operation: 0,10 s
- Lessons learned:
 - The eFA specification stresses the security features of currently available WS-stacks:
 - About 145 forum interactions between the Siemens development team for eFA security and the WSIT team at Sun over 6 months. About 10 bugs were reported.
 - We did not get the impression that other projects worldwide were burdening the properties of this WS-stack to the same degree as we had to do.



Conclusions

- Security is a cross-cutting concern in SOA. It requires to adapt new concepts and technologies including:
 - Identity federation
 - Attribute-based authorization
 - XML security technologies for WS-based SOA
- Do SOA security the SOA way:
 - Work-split: separate supply of security functionality from its usage
 - Self-containment: encapsulate security functionality in form of services
 - Re-use: use these security services in business application
 - Loose-coupling: decouple authorization from initial authentication
- The features of WS-stacks contribute to SOA security but do not solve the overall security problem.

Abbreviations

AOP	Aspect Oriented Programming
DAC	Discretionary Access Control
eFA	Elektronische Fallakte (engl.: ECR – Electronic Health Record)
IdP	Identity Provider
JAX-WS	Java API for XML-based Web Services
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PoP	Proof-of-Possession
PMA	Policy Management Authority
RP	Resource Provider
SAML	Security Assertion Markup Language
SOA	Service-Oriented Architecture
SSO	Single Sign On
STS	Security Token Service
WCF	Windows Communication Foundation
WS	Web Services
WSDL	Web Services Description Language
WSFED	Web Services Federation
WSIT	Web Services Interoperability Technologies
WSSE	Web Services Security Extensions
XACML	eXtensible Access Markup Language

Author

Dr. Oliver Pfaff
Siemens AG
Med GS SEC DI 1
E-Mail: oliver.pfaff@siemens.com