MTR100431

MITRE TECHNICAL REPORT

MITRE

Integrating Distributed Services Securely

Use Cases and Integration Templates

Gerald Beuchelt
November 2010

This page intentionally left blank.

MTR100431

MITRE TECHNICAL REPORT

MITRE

Integrating Distributed Services Securely

Use Cases and Integration Templates

Gerald Beuchelt
November 2010

Sponsor: ESC
Dept. No.: E111

Bedford, MA

Approved By:

Susan Borgeson	November 2, 2010
Name and Title of Approval Signature	Date

Table of Contents

List of Figures

List of Tables

This page intentionally left blank.

1 Introduction

This paper documents technical work conducted for the Electronic System Center (ESC) at Hanscom AFB, to create a description of the minimal requirements for a web service security infrastructure. The intent is for the infrastructure to be used by programs and projects that are moving towards distributed service architecture. To achieve this and make the results applicable to a wide array of programs, we analyzed the technical requirements of a number of programs of records (PORs) and distilled common use cases for service chaining, as well as typical service integration requirements.

Going forward, this paper is intended to be a living document: as technologies and requirements change, this paper should be updated.

2 Trust for Distributed Services

2.1 Distributed Security

For any POR moving from a tightly coupled architecture to more open service architecture, security is a big challenge. Not only is the technology stack significantly different, but also the decoupling of functionality into a large number of component service results in a significantly broader attack surface. With the significant resource requirements of a comprehensive distributed security system, the risk assessment and the cost/benefit determination becomes a major challenge.

While a provably secure system is technically and economically not feasible, even a highly secure system performance impacts may render the overall system unusable. For example, a fully orchestrated, WS-Security mediated 2 or more step service chain would result in latencies that are not acceptable: IBM published a paper[1] on the impact of WS-Security (even with XML Encryption and Signature turned off) that indicated a massive performance loss from turning WS-Security on.

As such, any tenable distributed architecture must perform a trade-off of security and performance needs.

2.2 Establishing Trust Domains

One way to balance security and performance is to avoid a full authentication and access decision for transactions that are performed in the same "trust domain". In lieu of a formal definition, as trust domain can be understood as a collection of services and servers that are logically and/or physically close to each other and trust the other (or a common entry point) with their access decision. In a way, a trust domain may become a "sub-enclave" that communicates with the outside in a very different way than with each other. Two principal approaches are possible:

1. Dedicated entry point. In this approach, all communications with the trust domain are performed through a well-defined entry point. In this case, only the entry point (which may be a specialized device or service) performs access control decisions including authentication and authorization of requests. All systems behind the entry point have complete

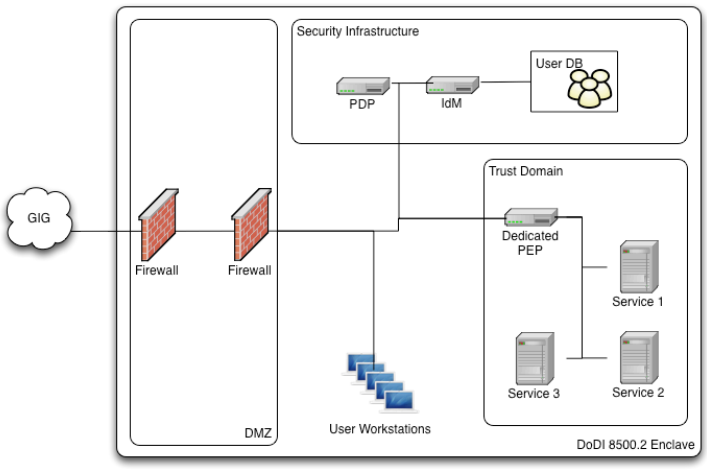

Figure 2-1: High level enclave view

[1] "Java Web services: The high cost of (WS-)Security", Dennis Sosnoski, IBM developerWorks, 07 Jul 2009, available at http://www.ibm.com/developerworks/java/library/j-jws6/

trust in that decision. Typically, the components services would not communicate directly with any entity outside the trust domain, making the entry point effectively a security firewall. In fact, the dedicated Policy Enforcement Point (PEP) may be implemented through a specialized XML Firewall or similar device.

2. Distributed access. For this approach, some (or all) services within the trust domain implement access control enforcement and allow communication with the outside. However, any inside communication would still be without further checks, since the services trust each other's access control system.

There are benefits and drawbacks to both approaches. The dedicated entry point allows a very strict configuration control, but might turn into a bottleneck for high-traffic trust domains. In turn, distributed access will be more scalable, but harder to control. Hybrid models (including load balancing techniques) may be used in some deployments.

2.3 Chaining Use Cases and Trust

Section 3.1 identifies a number of service chaining use cases. These use cases capture a small number of common web service interactions that any security infrastructure for distributed service environments should address. Specifically, they define scenarios where services are "chained", i.e. where a service or web application (the "intermediary service") invokes another service ("Chained" service) to complete the request.

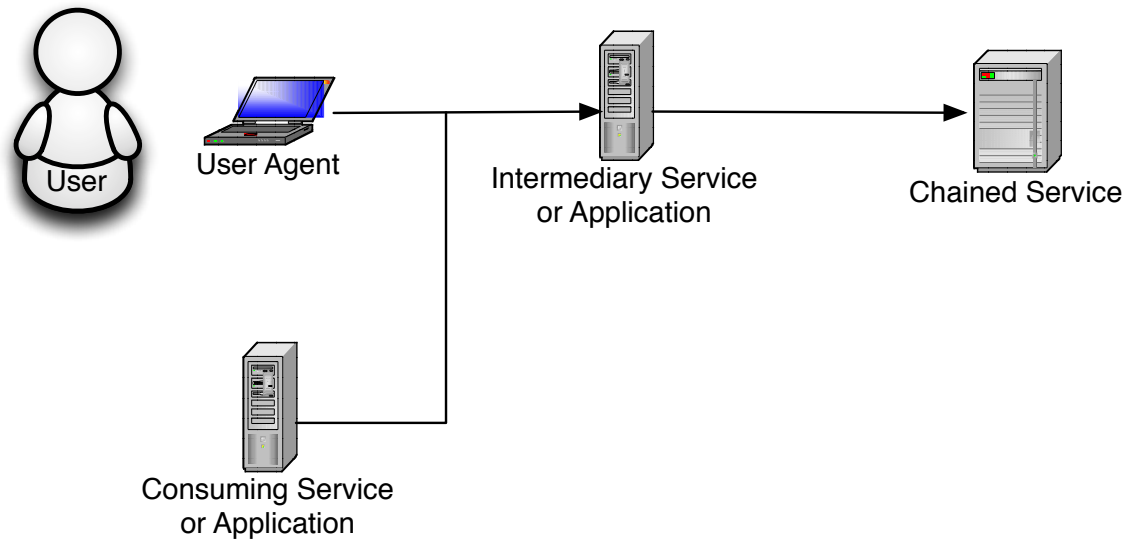

Figure 2-2: Basic service chaining

Implicit in these use cases is the assumption that chained services may trust the intermediary service – or not. This is a different level of trust described in the section on trust domains: even if a chained service trusts the intermediary to have authenticated the end-user (or machine consumer), it would still require to get the entity's identity from the intermediary and perform its own additional access checks. If it does not trust the intermediary's authentication, a full authentication and authorization check is required. The following table summarizes the classification of these use cases:

Trust Level	Authentication	Authorization
No Trust	At relying service	At relying service
Service chaining with partial trust	At intermediary ("sender vouches")	At relying service
Trust domain	At intermediary (or dedicated entry point)	At intermediary

Table 1: Trust Levels

It should be noted that this model is intentionally simplistic: it only addresses a "next-neighbor" trust chain, where a relying service may only delegate responsibility for authentication or authorization to an immediate partner (either an intermediary service, or a centralized access manager/Policy Decision Point (PDP)). In a more complex scenario, the relying service would be able to explicitly allow or deny the intermediary to further delegate the trust operation.

2.4 Summary of Trust Patterns

The discussion above identifies a number of independent dimensions that are needed to categorize the approach. These are:

- Pattern scale. This parameter is described in more detail in a forthcoming public MITRE paper. It identifies at which architectural level (Internal, Trust domain, Enclave, Super-Enterprise) a trust relationship exists. Depending on the pattern scale, the management of the trust relationship will vary.
- External interaction model. Whether there are many independent access points into a (sub-)enclave or a single one is described through this parameter. It should also capture the level of centralized configuration control over these endpoints.
- Trust delegation. Services may delegate authentication or authorization.

2.5 Using Security Integration Templates

The high-level security integration templates identified in section 3.2 may be used to integrate existing individual services or trust domains into common web service security infrastructure. Any design implementing these templates should indicate to what extent their implementation of these patterns is supportive of the specific use case.

3 Technical Requirements for Operational Systems

The requirements in this section are intended to address the most urgent need of distributed service architectures. They are – quite intentionally – not comprehensive in nature, but attempt to take an "80/20" approach.

3.1 Service Chaining Use Cases

This section identifies eleven service chaining use cases that address a number of possible situations that may occur in distributed service architectures. These use cases differentiate between Remote Procedure Call (RPC)/SOAP and RESTful/HTTP architectural styles for clarity only. The use cases do not exhaust the possible variation over trust parameters identified above, but are rooted in service chaining situations that are commonly found in operational systems.

3.1.1 Use Case 1

An operator accesses a web application that provides the entire service without needing to access any further data store – or – access to any further data store does not require authentication, authorization, and accounting (AAA). Authentication and authorization to the web application must be verified; access must be logged.

3.1.2 Use Case 2

An operator accesses a web application that needs to access a non-web service data store (LDAP directory, SMB/CIFS based file server, RDBMS). Authentication and authorization to the web application must be verified; access must be logged. Access to the non-web service data store must be authenticated, authorized, and/or logged. For access to the non-web service data store, credentials for the web application (authentication information or certified attribute information) are required.

3.1.3 Use Case 3

An operator accesses a web application that needs to access a non-web service data store (LDAP directory, SMB/CIFS based file server, RDBMS). Authentication and authorization to the web application must be verified; access must be logged. Access to the non-web service data store must be authenticated, authorized, and/or logged. For access to the non-web service data store, credentials for the end-user (user authentication information or certified attribute information) are required.

3.1.4 Use Case 4

An operator accesses a web application that needs to access a SOAP web service. Authentication and authorization to the web application must be verified; access must be logged. Access to the SOAP web service must be authenticated, authorized, and/or logged. For access to the SOAP web service, credentials of the web application (authentication information or certified attribute information) are passed to the SOAP web service according to the "service chaining with partial trust" trust level where the web application is the intermediary.

3.1.5 Use Case 5

An operator accesses a web application that needs to access a SOAP web service. Authentication and authorization to the web application must be verified; access must be logged. Access to the

SOAP web service must be authenticated, authorized, and/or logged. For access to the SOAP web service, credentials of the end-user (user authentication information or certified attribute information) are passed to the SOAP web service according to the "service chaining with partial trust" trust level where the web application is the intermediary.

3.1.6 Use Case 6

An operator accesses a web application that needs to access a HTTP web service using a RESTful architectural style. Authentication and authorization to the web application must be verified; access must be logged. Access to the HTTP web service must be authenticated, authorized, and/or logged. For access to the HTTP web service, credentials of the web application (authentication information or certified attribute information) are passed to the HTTP web service according to the "service chaining with partial trust" trust level where the web application is the intermediary.

3.1.7 Use Case 7

An operator accesses a web application that needs to access a HTTP web service using a RESTful architectural style. Authentication and authorization to the web application must be verified; access must be logged. Access to the HTTP web service must be authenticated, authorized, and/or logged. For access to the HTTP web service, credentials (user authentication information or certified attribute information) of the end-user are passed to the HTTP web service according to the "service chaining with partial trust" trust level where the web application is the intermediary.

3.1.8 Use Case 8

An operator accesses a rich client application that needs to access a SOAP web service. Authentication and authorization to the rich client application must be verified; access must be logged. Access to the SOAP web service must be authenticated, authorized, and/or logged. For access to the SOAP web service, credentials of the end-user (user authentication information or certified attribute information) are passed to the SOAP web service according to the "service chaining with partial trust" trust level where the web application is the intermediary..

3.1.9 Use Case 9

An operator accesses a rich client application that needs to access a HTTP web service using a RESTful architectural style. Authentication and authorization to the rich client application must be verified; access must be logged. Access to the HTTP web service must be authenticated, authorized, and/or logged. For access to the HTTP web service, credentials (user authentication information or certified attribute information) of the end-user are passed to the HTTP web service according to the "service chaining with partial trust" trust level where the web application is the intermediary..

3.1.10 Use Case 10

An operator accesses a web application or rich client application that needs to access a SOAP web service. This SOAP web service needs to invoke another service (non-web service data source, SOAP or HTTP web service using a RESTful architectural style). Authentication and authorization to the first service must be verified; access must be logged. Access to the second service (non-web service data source, SOAP or HTTP web service) data store must be authenticated, authorized, and/or logged. For access to the second service, credentials (authentication information or certified attribute information) of the first SOAP web service are

passed to the SOAP or HTTP web service according to the "service chaining with partial trust" trust level where the web application is the intermediary..

3.1.11 Use Case 11

An operator accesses a web application or rich client application that needs to access a SOAP web service. This SOAP web service needs to invoke another service (non-web service data source, SOAP or HTTP web service using a RESTful architectural style). Authentication and authorization to the first service must be verified; access must be logged. Access to the second service (non-web service data source, SOAP or HTTP web service) data store must be authenticated, authorized, and/or logged. For access to the second service, credentials (user authentication information or certified attribute information) of the end-user are passed to the SOAP or HTTP web service according to the "service chaining with partial trust" trust level where the web application is the intermediary..

3.2 Security Integration Templates

In order to accommodate for the wide range of different types of services and their internal security architectures, this section identifies a number of templates for integrating into a common web service security infrastructure. The idea is to create a central instance of basic security components such as an Identity Manager, a PDP, etc. and offer a catalog of pre-defined templates to integrate an existing service with this core security infrastructure.
Throughout this section we will indicate divergences in the requirements found in the programs and projects we talked to. The core requirements for the security infrastructure and the integration patterns were common to most projects we included in the compilation of this list. For those elements marked OPTIONAL, only a small number of interviewees indicated that they would need support for the element in question.

3.2.1 Requirements for the Common Infrastructure

A common security infrastructure for web service security should provide the following capabilities:

- An Access Manager for centralized access control to all capability services that are integrated with infrastructure
    - Provides central authorization policy store **OR** integrates with Policy Information Point
        - Policies can be written based on local and enterprise attributes (ABAC)
        - ABAC policies can be used to effectively implement role based authorization control (RBAC)
        - Policies can be imported and exported in XACML (OPTIONAL)
    - Open architecture for integrating different user management databases
        - Microsoft Active Directory (OPTIONAL)
        - RDBMS user store (OPTIONAL)
        - LDAP Directory using inetOrgPerson or extension thereof (OPTIONAL)
        - Other (OPTIONAL)
- User provisioning system
    - Automated user provisioning and deprovisioning through administrator interface
    - Automated user provisioning and deprovisioning through Java API (OPTIONAL)
    - Automated user provisioning and deprovisioning through web service API (OPTIONAL)
    - Automated user provisioning and deprovisioning through SPML (OPTIONAL)

- Federation support
  - Allow capability provider service (service provider) to run autonomous user account database
  - Support account federation through WS-Trust/WS-Federation or ID-WSF (OPTIONAL)
  - Support account federation through other protocols (OPTIONAL)
- Centralized logging infrastructure
  - Supports flexible log management of all authentication and authorization interactions
  - Allows to integrate log messages from services

3.2.2 Interaction template 1: Security aware SOAP web service

- SOAP web service
- Uses common SOAP-based security protocols for authentication (e.g. WS-Security)
- May rely on external Policy Decision Point for authorization or perform internal authorization decision
- Uses XACML for PEP/PDP external communication (OPTIONAL)
- Can obtain identity of user or service that invokes service and log it
- Integrates into centralized logging infrastructure (OPTIONAL)

3.2.3 Interaction template 2: Security aware HTTP web service

- HTTP web service using RESTful architectural style
- Uses common HTTP security protocols for authorization (e.g. OAuth)
- Can obtain identity of user or service that invokes service and log it
- Integrates into centralized logging infrastructure (OPTIONAL)

3.2.4 Interaction template 3: Container based security/Container security agent

- Application runtime container (e.g. JEE application server) performs authentication and authorization services (e.g. through container based agent)
- Supports SOAP web services and web applications
- Supports HTTP web services using RESTful architectural style
- Can obtain identity of user or service that invokes service and log it
- Relies on external Policy Decision Point for authorization
- Uses XACML/OAuth for access decision (OPTIONAL)
- Integrates into centralized logging infrastructure (OPTIONAL)

3.2.5 Interaction template 4: Appliance mediated access control

- Uses external appliance (either hardware or virtual appliance) for access control
- Supports SOAP web services, web applications
- Supports HTTP web services using RESTful architectural style
- Supports other transports such as FTP, JMS, etc. (OPTIONAL)
- Relies on external Policy Decision Point for authorization
- Can obtain identity of user or service that invokes service and log it
- Uses XACML/OAuth for policy decision (OPTIONAL)
- Integrates into centralized logging infrastructure (OPTIONAL)

3.2.6 Interaction template 5: Reverse security proxy (OPTIONAL)

- Supports legacy applications and services by translating web service security tokens and authorization decisions into legacy access tokens
- Can obtain identity of user or service that invokes service and log it
- Integrates into centralized logging infrastructure

4 The Broader Picture

4.1 Integration with Authorization Focused Efforts

There has recently been strong emphasis on patterns and templates necessary for authorization. While these are necessary for creating a comprehensive Attribute Based Authorization Control (ABAC) regime, there is little guidance on where credential verification and token issuance should occur and how identity information can flow within or across trust boundaries. This paper does not address all of these issues, but it provides a concrete set of template and uses cases to verify the architecture against. With its focus on identity propagation through service chaining and system-to-system level integration templates, the requirements for a distributed service architecture identified in this paper are thus complementary to the efforts of authorization centric approaches.

4.2 Re-Use for Other Distributed Systems

Since this paper outlines a more detailed focus on the identity management aspect necessary for orchestrating services more securely, it can be seen as a building block for all types of distributed architectures. Depending on the goals and policy constraints many of these systems the content of this paper may be useful in its entirety or only portions. This applies to the trust discussion, the chaining use cases, as well as the suggested security integration templates.

5 List of acronyms

Acronym	Meaning
AAA	Authentication, Authorization, Accounting
ABAC	Attribute Based Access Control
CIFS	Common Internet File System protocol
DMZ	De-Militarized Zone
HTTP	Hyper Text Transport Protocol
ID-WSF	Liberty Web Service Framework
IdM	Identity Manager
IdP	Identity Provider
JEE	Java Enterprise Edition
LDAP	Lightweight Directory Access Protocol
PDP	Policy Decision Point
PEP	Policy Enforcement Point
POR	Program of Record
RBAC	Role Based Access Control
RDBMS	Relational Database Management System
RPC	Remote Procedure Call
SMB	Server Message Block protocol
SOAP	Formerly Simple Object Access Protocol
SPML	Service Provisioning Markup Language
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

Table 2: List of acronyms

