Security as a Service Model in SOA

JULIANA GEORGIEVA, MARIANA GORANOVA
Department of Programming and Computer Technologies
Technical University of Sofia
Sofia, Bul. "Kl. Ohridski" 8, bl. 2
BULGARIA
july@tu-sofia.bg, mgor@tu-sofia.bg

Abstract: The software architecture requires interoperable security mechanisms. This article focuses on applying security requirements to service-oriented solution design. SOA security is very much concerned with what the system is supposed to do and what can go wrong. This article presents the service-oriented approach — security services that can be developed and tested and applied against many types of applications or scenarios. The proposed concept has the contribution to allow for SSAS (Software Security as a Service) providers to provide access to software services without requiring the customer to host this service within their local environment. In this model, the access control decision and (ideally) enforcement functionality is not embedded within an application. The split of enforcement and decision point has its advantages.

Key-Words: - SOA, Service, ESB, SSAS, Security, Security Enforcement Service, Security Decision Service

1 Introduction

In SOA consumers can dynamically locate service producers. The inherent benefit of SOA is the loose coupling between the producer and the consumer, which eases the construction of component based solutions. The model of ESB (Enterprise Service Bus) supplies a communication layer to support service interactions (consumer – services). To preserve loose coupling, security must also be implemented as a service - to avoid tightly bound security and thereby tight binding of the services themselves. An ESB must ensure that the integrity and confidentiality of the services that it carries are maintained. The services must integrate with the existing security infrastructures to address the essential security functions. The ESB can provide security either directly or by integrating with other security components.

"SOA Security is two separate things, and solves two separate problems - securing SOA-based infrastructures, and applying SOA principles to security" [2]. Too many SOA Security articles focus only on the first meaning of SOA Security (making SOA more secure) than on the second (applying SOA principles to security to make it easier to deploy and manage).

"SOA-flavored Security means making security more management and easy to deploy by isolating re-usable components of security and providing them as managed services" [2]. For example, the OASIS DSS standard explains how digital signature services can be used in order to provide signing and signature validation services over the network accessed using a Web Services interface. This provides a good framework for key management. Similarly, specifications such as XKMS (XML Key Management Specification), XACML (eXtensible Access Control Markup Language), and WS-Trust are all about applying SOA to security, to solve interoperability problems. The handling of authorization of digital identities in a SOA is presented in [5] creating a design pattern for the integration of legacy systems with SOA using out-of-the-box (unmodified) application servers. Applying security practices to service-oriented solution design with an emphasis on considerations raised by authentication, authorization, auditing, and assurance is depicted in [4]. A framework for security-oriented - software service composition and evolution is given in [6]. Key building blocks of the framework are a semantic model for specifying the security objectives and properties at the service and system levels.

In SOA the primary security functions could correspond to processing functionality provided by reusable utility services, so called "security services". These security services are needed to mediate communication between a subject and its objects (service provider and its consumers).

The proposed concept has the following main contribution - the “Software Security as a Service” (SSAS) model allows for SSAS providers to provide access to software services without requiring the customer to host this service within their local environment. By analogy, SSAS occurs when an application (which may in turn be a service) does not internalize, or locally host, security functionality. This, logically, is the model that has been adopted by the International Standards Organization (ISO) model of security as implemented through a “Policy Decision Point” (PDP) and a “Policy Enforcement Point” (PEP) [ISO10181]. The PDP is the entity responsible for the access control decision required to allow/deny access to a resource. The PEP is the entity responsible for enforcing this decision, appropriately allowing or denying the access in response to the access control decision. In a security as a service model, the access control decision and (ideally) enforcement functionality is not embedded within an application. Instead, the application often provides enforcement functionality for an external access control decision but relies on a service to acquire the decision. This split of enforcement and decision point has advantages, beyond moving the security complexity out of the realm of the application developer. This separation of concerns approach allows for multiple enforcement points to re-use the same decision point functionality. This in turn promotes component re-use as well as the consistent application of access control decisions across an environment. Moving towards an SOA, this enforcement will in turn move to the service invocation layer, as described in this paper. Extending the separate decision/enforcement approach for an SOA, we can consider a Security Enforcement Service (SES) as a service based PDP, sharing the common characteristics of an ESB. An SES provides a single service that is easily replicated, scaled, and distributed across an environment to provide PDP functionality for all service enforcement points, regardless of the service’s actual instantiation.

2 Security as a Service - Background

2.1 Security Appliance Architecture

SOA security appliances implements securities as a service through a hardware-based gateway and XML proxy that can parse, filter, validate schema, decrypt, verify signatures, access-control, transform, and sign and encrypt XML message flows. The security appliances is a server side security gateway that allows for all keys and tokens used to provide integrity and confidentiality to services exposed through the gateway to be managed at one point, the appliance. That means that clients invoking any number of services exposed through this appliance need only be configured to trust keys or tokens from this single gateway, rather than keys and tokens from each service. The appliance needs to be configured to trust keys and tokens from each client. Security appliances should implement the following features: 1) Comprehensive security - all XML and web services security functions in one device; 2) Web services access control via new technologies (like SAML, XACML, WS-Security) to control access to applications; 3) Centralized configuration and policy management ; 4) Performance - purpose-built to secure without degradation; 5) XML-based agility - future-proof for changing standards, policies, partners; 6) Appliance-based - drop-in device secures multiple applications at once; 7) Easy integration - interoperates with and augments existing security systems; 8) XML transformation - includes XPath/XSLT acceleration features.

The multiple service consumers can be connecting to multiple service providers. The mediation between these flexible and dynamic connections is implemented by the ESB. Some of the requirements to provide security mediation are: 1) identity; 2) authorization; 3) confidentiality; 4) protection from attack of services; 5) audit of logged events for compliance.

2.2 SOA Security Requirements

There are given a number of SOA security requirements, gathered from the relevant literature [7], [8], [9], [10].

2.2.1 Information

2.2.1.1 Security Requirements for Stored Information

The following could be enumerated: Service requests must be authenticated [7]; Service consumer is authorized to a service (access control and service consumer authorization); Operation of a service consumer must be controlled (the privilege of the service consumers and information protection); Service provided information must be authenticated; Content/format of the message is valid (information verification).

2.2.1.2 Security Requirements for information in transmission

This group comprises the following: 1) Content of the exchanged message should be protected (confidentiality of a message in transmission);
2) Information receiver (service consumer and provider) must verify that the content of the received messages have not been modified during transmission (integrity of the message) [7]; 3) Content of the received message must be authenticated (authentication of the message); 4) Information sender and information receiver must be authenticated;

2.2.2 Service

2.2.2.1 Security Requirements for a Single Service

The security policies can be defined clearly. The policies can describe the kinds of contexts under which different operations provided by service can be executed; so security policy is constraint to the service operations [10].

The policies should be externally published so a service consumer can choose an appropriate service based on security policies specifications.

Service must satisfy security request of service consumer. If a service's consumers are uncertain, the security policies can be effectively changed.

Service definition should be defined clearly.

2.2.2.2 Security Requirements for service composition

The access restriction to the composed service is at least the sum of the restrictions of all underlying operations it is composed of and that are invoked compulsorily. This allows checking authorization at an earlier stage, thereby limiting unnecessary calls ending in rollback operations if particular permissions for invoked basic service operations are missing.

2.2.3 Organization

The following three requirements can ease to build trust between services: 1) Services in the system including their definitions must be authenticated [10]; 2) Services in the system must be authorized; 3) Service Level Agreement (SLA) must be specified clearly.

All interactions between services should be logged in order to identify potential attacks, trace the attackers, gather information of abnormal activities, and recover failed interactions (audit ability).

2.3 SOA Security Standards

The complex security requirements of SOA and specifically Web Services are driving a new set of security standards [7], [8], [9], [10], popularized by the SOA movement (see Table 1).

Security service	Standards
Identity Services	SAML (Security Assertion Markup Language), WS-Federation
Authentication Services	WS-Trust, SAML, Public Key Infrastructure (PKI)
Authorization Services	XACML (eXtensible Access Control Markup Language), WS-Federation
Audit Services	CBE (code building environment), WS-BaseNotification
Confidentiality and Integrity Services	WS-Security, WS-SecureConversation, XKMS

Table 1

3 Security as a service model

Security within an SOA needs to be seen as an incremental set of services that can be applied at different layers (message, transport , service invocation, service fulfillment) or at different control points (identity of consumers, their authentication, their authorization and privacy).

3.1 Typical architecture for an SOA application with security enforcement

The Web application/portal server leverages existing applications/services either directly or through an ESB. In the deployment architecture for an SOA (Fig.1) security is enforced at various points. The proxy can enforce confidentiality and integrity, identity, validation, authentication, and auditing. The identity derived at the proxy might need to be propagated to the application server where the propagated identity needs to be accepted and additional security checks, such as authorization, can be enforced, as well as auditing this activity. Further security enforcement can be performed by the other components within the architecture as well.

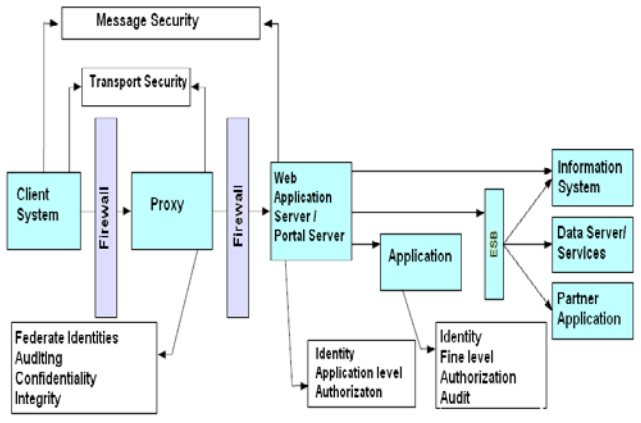
Message Security
Transport Security
Client System
Firewall
Proxy
Firewall
Web Application Server / Portal Server
Application
ESB
Information System
Data Server/ Services
Partner Application
Federate Identities Auditing Confidentiality Integrity
Identity Application level Authorizaton
Identity Fine level Authorization Audit

Fig.1

3.2 SOA Security services and some proposals for realization solutions

3.2.1 Identity Service

The identity of all consumers is represented with a unique value, or identifier, such as a user name or a UUID (universally unique identifier), creating an *identity token*, a unique representation of the identity and attributes of the consumer in a standardized format.

Standards for solutions: 1) SAML (Security Assertion Markup Language) is an XML framework for exchanging authentication and authorization information; 2) WS-Federation (Web Services Federation Language) is a specification to define mechanisms to allow different security realms to federate by allowing and brokering trust of identities, attributes, authentication between participating Web services.

3.2.2 Authentication Service

Authentication is the process of proving that the consumer legitimately has their claimed identity by evaluating additional information (*authentication credentials - passwords*). Authentication information might be bound to a request and carried with the request in the form of a *security token* (which includes additional information that allows it to be used as part of the authentication process). This is frequently a system-level service because it deals with the processing of system policies (such as password policies). This warrants a separate service because authentication logic is generally not valuable (or reusable) when intertwined with other application logic.

This is a standards-based service to handle which user identity is passed, and how is it passed. For example, *WS-Trust* defines a mechanism for security token exchange to manage trust relationships.

3.2.3 Authorization Service

It is the process of evaluating if an authenticated identity is allowed to have its request fulfilled. Privacy as a type of authorization is based on the data being retrieved where access to *Personally Identifiable Information* (PII) is controlled. Authorization is always enforced locally, close to the thing being protected. In SOA, this thing is the service provider. While coarse-grained authorization can be implemented at a global level, finer grained authorization requires mapping to the service and its operations.

Solutions: 1) From a design perspective, authorization should be viewed at both system and service levels (the latter always being enforced locally); 2) XACML (a policy language that can use SAML assertions) standard describes general access control requirements and a request/response language how to form a query to determine if a given action is allowed or not and how to interpret the result. In XACML, a policy enforcement point (PEP) intercepts a service request, gathers all relevant information and credentials about the request, and then passes them to a policy decision point (PDP) to render a decision on granting access. The basis of the decision is a three part rule where the resource that is attempting to be accessed is mapped to a condition and an action for a subject. XACML enables the authorization logic to be widely reused.

3.2.4 Audit Service

It provides support for the principle of accountability and detection of security-policy violations in distributed systems. So we need a way to generate audit events end-to-end for transactions, collate these into a common format, and allow real-time and post processing of events for reporting. We also need ways to verify if these events are compliant with policy. Audit services provide detection and response features that serve to answers questions around what digital subjects performed what actions to what objects.

Solutions: 1) CBE (code building environment) - is a control system with integrated make-like functions written in Java; 2) WS-BaseNotification is a web services specification which defines the interface WS-Notification (a group of specifications related to the WS-Resource framework) that allows event driven programming between web services clients (consumers) and servers (producers).

3.2.5 Confidentiality and Integrity Service

Identity is the basis of all access control decisions, and most of these decisions generally begin with authenticating the request. There are already numerous directory services for any given application, already authenticate the user client. So the question is: how does the system reuse the information from the authentication events that have occurred? The answer might be: establishing federation between policy decision points within the service requester and service provider.

Solutions: 1) The WS-Security specification provides end-to-end message-level security; 2) WS-SecureConversation defines extensions that build on WS-Security to provide secure communication. Specifically, it defines mechanisms for establishing and sharing security contexts, and deriving session keys from security contexts; 3) XKMS (XML Key Management Specification) – protocols with their two parts (XML Key Information Service Specification and XML Key Registration Service Specification) for distributing and registering public keys, suitable for use in conjunction with the proposed standard for XML Signatures.

3.3 Infrastructure Managed Security Model

A Security Enforcement Service, or SES, is a potential component of an ESB enabled SOA architecture. The SES allows access control decisions to be applied to a request to access a service, whether that service is implemented as a CICS (Customer Information Control System**)** resource, a .Net resource, or a J2EE resource. Within the context of SOA and ESB, a SES is independent of the transport method of a particular message/service request, as shown in Fig. 2, below. A reverse proxy moves authentication functionality to the edge of the network, so that only authenticated users are allowed into the Trusted Network. This additional step consolidates security functionality to a single logical point (the reverse proxy server) thus identifying a common security point. This common security point provides an opportunity to define a common security service, a core component of SOA. A common security service (Security Decision Service) can also help with systems management issues eliminating the need for detailed knowledge of authorization decisions in the back-end application. This approach also allows for some level of platform independence. Since the authentication is now done at the edge, it doesn't matter if the service is implemented as a type X resource or a type Y resource since the reverse proxy has determined user access to the resource based on the message request and the authentication context. Similarly, a reverse proxy can provide some addressing transparency (another characteristic of SOA) through "routing" by mapping between the internal architecture and a publicly requested URL. The security attributes such as maintainability, auditability and reliability cannot be represented by information security (it is a view of SOA security). There is need of modelling quality attributes from different viewpoints such as information view, service view, and organization view.

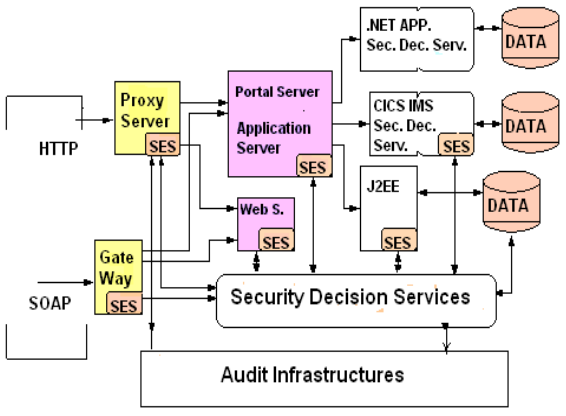

Fig. 2

3.4 A Framework for a Secure Decision Services Composition

3.4.1 Definition of security objectives and properties

A group of high-level security objectives (like integrity, authenticity, confidentiality, authorization) and detailed security properties (passwords, keys – private, public, secret, digital signatures) could be defined. We could distinguish: 1) system-level security objectives, which are defined with the system architecture; 2) service-level security objectives, associated with specific service functionalities; 3) service-level security properties, required to achieve the security objectives.

3.4.2 Security Service Composition

At this stage a few considerations have to be taken: 1) select a collection of candidate services that can potentially satisfy the security objectives of the overall system (each service might have a number of security properties that, individually or in combination with others, can fulfil its security objectives); 2) the choice between the services carried out through the iterative exchange of the preferred security properties supported by the candidate services in a composition and governed by a negotiation protocol that specifies the rules of interaction between the services; 3) verifying the security properties of the composite system against the systems-level security objectives.

3.4.3 Additional system security requirements and the adaptation of services

The system evolution has to be conformed with: 1) adapt an existing service contract under different terms; 2) establish a service contract with a newly selected service; 3) change the existing composition architecture, and consequently instantiate it by adapting certain existing service contracts and establishing new contracts with new services.

4 Conclusion

Our approach to security-oriented service composition and evolution is different from traditional approaches to system security. To preserve loose coupling, security must also be implemented as a service - to avoid tightly bound security and thereby tight binding of the services themselves. The focus is on applying SOA principles to security.

The use of common IT Security Services enables a consistent security implementation. It also minimizes development and deployment costs for these security services and for the SOA environment on which these security services are reused. A general view on the different kind of services and some ways of their solutions (comprising the known already standards and not only them) are given. The proposed in this paper model has the contribution to allow for SSAS (Software Security as a Service) providers to provide access to software services without requiring the customer to host this service within their local environment. A framework for a secure decision services composition is given.

ACKNOWLEDGMENT
This paper is supported by the National Scientific Fund of Ministry of Education and Science in Bulgaria under the contract № DO 02-175/2008.

References:

[1] Axel Buecker, Paul Ashley, Martin Borret, Ming Lu, Sridhar Muppidi, Understanding SOA Security, Design and Implementation, IBM Thechnical Support Organization, 2007.

[2] Joe McKendrick, SOA security: isn't SOA itself a security solution?, http://www.zdnet.com/blog/service-oriented/soa-security-isnt-soa-itself-a-security-solution/2728.

[3] Siming Kou , Muhammad Ali Babar, Amit Sangroya, Modeling Security for Service Oriented Applications, ECSA, 2010, ACM

[4] Gunnar Peterson, Security in SOA - It's the Car, Not the Garage, SOA Magazine Issue XV: February 2008

[5] Christian Emig, Heiko Schandua, Sebastian Abeck, SOA-aware Authorization Control, Proceedings of the International Conference on Software Engineering Advances (ICSEA'06), 2006

[6] Jun Han, Ryszard Kowalczyk, Khaled M. Khan, Security-Oriented Service Composition and Evolution, XIII ASIA PACIFIC SOFTWARE ENGINEERING CONFERENCE (APSEC'06), 2006

[7] Delessy, N., A., and Fernandez, E., A. (2008) 'A Pattern-Driven Security Process for SOA Applications', Third International Conference on Availability, Reliability and Security, 416-421.

[8] Han, J., Kowalczyk, R. and Khan, K., M. (2006) 'Security-Oriented Service Composition and Evolution', Software Engineering Conference, 71-78.

[9] Lang, U. and Schreiner, R. (2007) 'Model Driven Security for Agile SOA-Style Environments', Securing Electronic Business Processes, Vieweg (2007), 147-156.

[10] Mouelhi, T., Fleurey, F., and Baudry B. (2008) 'A Generic Metamodel For Security Policies Mutation', IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW'08).