**Service Oriented Architecture (SOA) Security**

**Troy L Hutchison**

**CS526**

**DR Chow**

**Sunday, May 09, 2010**

**Service Oriented Architecture (SOA) Design**

**Introduction to SOA and security**

SOA is a business based approach to solving a business Information Technology (IT) problem versus a technical approach. SOA is a new approach to solving business IT problems because SOA focus is on the problem from a business prospective. In the past it has been very challenging to resolve business problems because of the limitations imposed by the software solution. SOA attempts to break the limitations and set businesses free to solve problem by not focusing on the how but the what. To accomplish this, the System Architects must first work with the Business Analyst to do a complete business process modeling analysis and create the problem statement. Then the system requirements must then be decomposed to testable components. Then the next step is to evaluate the current legacy silo systems to determine if and how they can be adapted to meet the business needs without compromising security. Through the use of Web Services the development team can create business services from business applications.

When we increase the complexity of our systems by integrating the many separate loosely coupled components into a single complex enterprise system we also introduce additional security risks.

**Getting Started with SOA**

**Business Process Modeling (BPM)**

Because we and trying to solve a business problem we must first have a clear understanding of the business its self before we bring an IT solution to solve a yet unknown problem statement. To accomplish this we must complete a business process analysis. This analysis will help us by clearly defining the problem or problems we are trying to solve before we determine the best technical approach to solving the problem. To aid the system architects there and many Business Process Modeling tools available.

After the analysis is completed the systems architect team conducts a survey to determine if there are any current legacy systems available that could be reused in order to solve the business problem. The BPM tool checks the SOA registry to identify any business functions that are already published. The legacy systems must be evaluated to determine the benefit and security risks involved in including them in the implemented system. The Business process modeling tool has the ability to logically link together business functions from existing applications. It enables a developer to link them together to make composite applications or slot them in at the appropriate point in the overall workflow. The BPM tool stores this information in the SOA component called the registry.

The BPM also enables the creation of new business functions. A developer may add whole new business functions or may simply add logic to run before or after an existing business function. In order to do this, the BPM tool includes a way of specifying a software process. When a new business function is created, the BPM tool adds the function's details to the SOA registry, including information about how it links to other components.

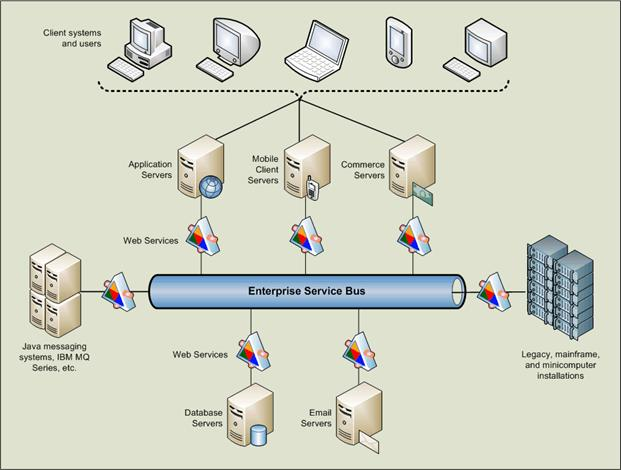
Using the BPM tool, business analysts design process flows and specify the movement of work from one person to another within a business process, linking in the applications that they need to use for the tasks that they have to carry out. It programs the workflow engine to carry out the business process.

**Web Services and SOA**

SOA and Web Services are two different things, but web services are the preferred standards-based way to implement SOA. Web services are software systems designed to support interoperable machine-to-machine interaction over a network. This interoperability is gained through a set of XML-based open standards, such as WSDL and SOAP. These standards provide a common approach for defining, publishing, and using web services. XML (eXtensible Markup Language) is a special language that enables programmers to define data in a way that any program can understand, WSDL (Web Services Description Language) is a XML-based language that provides a model for describing Web services, and SOAP (Simple Object Access Protocol) is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It relies on Extensible Markup Language (XML) as its message format and usually relies on other Application Layer protocols

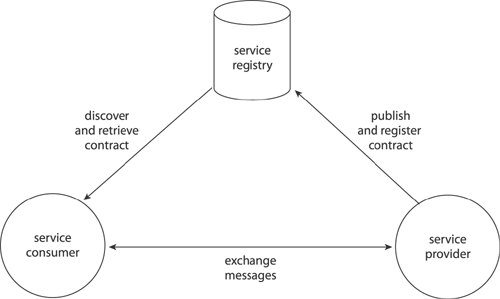
**The components of a SOA environment**

The components of a SOA environment consist of the Enterprise Bus, SOA Registry, Workflow engine, Service Broker, and SOA Supervisor.

****

**Enterprise Service Bus (ESB)**

The Enterprise Service Bus (ESB) is the backbone of the SOA infrastructure, providing standards-based, event-driven, and messaging-based interoperability between applications. The ESB provides mediation facilities between these applications to achieve interoperability. It also serves as the foundation upon which these services can be orchestrated and choreographed to build more complex applications while retaining the benefits of loose coupling and reusable assets. ESBs are designed to act as intermediaries between the SOA components, infrastructure services, and business processes. ESBs were not designed specifically to act as SOA intermediaries, but because a service oriented architecture needs a devoted intermediary in order to scale up for large numbers of users. The ESB hub and spoke topology allows for consistent approaches to security, centralized configuration management, and management of quality of service. The way ESB implements services, provides interface mapping, and adapters to existing systems. The ESB provides the framework for service reuse.



**SOA Registry**

A SOA registry is a central repository of business services available in the domain. The SOA registry stores information about each business service that has been approved for use in the enterprise. It also maintains the history of the service.

**Workflow Engine**

The workflow engine is a software component designed to connect a whole business process from end to end, flowing work from one individual or process to another until the entire business process is carried out.

**Service Broker**

Because SOA is loosely coupled the Service Broker is required to make the connection between the independent services. The Service Broker refers to the SOA Registry to make these connections.

**SOA Supervisor**

The SOA supervisor is the central point of control responsible for all SOA orchestration. The SOA supervisor interacts with the Infrastructure Services. If any of the components in the end-to-end service have any performance problems, the SOA supervisor sends the details to the appropriate Infrastructure Services, and the Infrastructure Services try to fix the problem.

**Eight Service Oriented Architecture Design Principals**

The eight design principles documented in SOA: Principles of Service Design provide rules and guidelines that help determine exactly how solution logic should be decomposed and shaped into distributable units. These principles reveal what design characteristics these units should have to be classified as "quality" services capable of fulfilling the vision and goals associated with SOA and service-oriented computing.

**Service Contracts (Standardization and Design)**

Services express their purpose and capabilities via a service contract. The Standardized Service Contract design principle is the most fundamental part of service-orientation because it requires that specific considerations be taken into account when designing a service’s public technical interface and assessing the nature and quantity of content that will be published as part of a service’s official contract. A great deal of emphasis is placed on specific aspects of contract design, including the manner in which services express functionality, how data types and data models are defined, and how policies are asserted and attached. There is a constant focus on ensuring that service contracts are both optimized, appropriately granular, and standardized to ensure that the endpoints established by services are consistent, reliable, and governable.

**Service Coupling (Intra-Service and Consumer Dependencies)**

Coupling refers to a connection or relationship between two things. A measure of coupling is comparable to a level of dependency. This principle advocates the creation of a specific type of relationship within and outside of service boundaries, with a constant emphasis on reducing (“loosening”) dependencies between the service contract, its implementation, and its service consumers. The principle of Service Loose Coupling promotes the independent design and evolution of a service’s logic and implementation while still guaranteeing baseline interoperability with consumers that have come to rely on the service’s capabilities. There are numerous types of coupling involved in the design of a service, each of which can impact the content and granularity of its contract. Achieving the appropriate level of coupling requires that practical considerations be balanced against various service design preferences.

**Service Abstraction (Information Hiding and Meta Abstraction Types)**

Abstraction ties into many aspects of service-orientation. On a fundamental level, this principle emphasizes the need to hide as much of the underlying details of a service as possible. Doing so directly enables and preserves the previously described loosely coupled relationship. Service Abstraction also plays a significant role in the positioning and design of service compositions. Various forms of meta data come into the picture when assessing appropriate abstraction levels. The extent of abstraction applied can affect service contract granularity and can further influence the ultimate cost and effort of governing the service.

**Service Reusability (Commercial and Agnostic Design)**

Reuse is strongly emphasized within service-orientation; so much so, that it becomes a core part of typical service analysis and design processes, and also forms the basis for key service models. The advent of mature, non-proprietary service technology has provided the opportunity to maximize the reuse potential of multi-purpose logic on an unprecedented level. The principle of Service Reusability emphasizes the positioning of services as enterprise resources with agnostic functional contexts. Numerous design considerations are raised to ensure that individual service capabilities are appropriately defined in relation to an agnostic service context, and to guarantee that they can facilitate the necessary reuse requirements.

**Service Autonomy (Processing Boundaries and Control)**

For services to carry out their capabilities consistently and reliably, their underlying solution logic needs to have a significant degree of control over its environment and resources. The principle of Service Autonomy supports the extent to which other design principles can be effectively realized in real world production environments by fostering design characteristics that increase a service’s reliability and behavioral predictability. This principle raises various issues that pertain to the design of service logic as well as the service’s actual implementation environment. Isolation levels and service normalization considerations are taken into account to achieve a suitable measure of autonomy, especially for reusable services that are frequently shared.

**Service Statelessness (State Management Deferral and Stateless Design)**

The management of excessive state information can compromise the availability of a service and undermine its scalability potential. Services are therefore ideally designed to remain stateful only when required. Applying the principle of Service Statelessness requires that measures of realistically attainable statelessness be assessed, based on the adequacy of the surrounding technology architecture to provide state management delegation and deferral options.

**Service Discoverability (Interpretability and Communication)**

For services to be positioned as IT assets with repeatable ROI they need to be easily identified and understood when opportunities for reuse present themselves. The service design therefore needs to take the “communications quality” of the service and its individual capabilities into account, regardless of whether a discovery mechanism (such as a service registry) is an immediate part of the environment.

**Service Composability (Composition Member Design and Complex Compositions)**

As the sophistication of service-oriented solutions continues to grow, so does the complexity of underlying service composition configurations. The ability to effectively compose services is a critical requirement for achieving some of the most fundamental goals of service-oriented computing.

Complex service compositions place demands on service design that need to be anticipated to avoid massive retro-fitting efforts. Services are expected to be capable of participating as effective composition members, regardless of whether they need to be immediately enlisted in a composition. The principle of Service Composability addresses this requirement by ensuring that a variety of considerations are taken into account.

**Security threats to SOA**

Some of the most critical threats to a SOA environment are Confidentiality, Integrity and Availability of the information being delivered through the Web Services.

The purpose of SOA is to support business systems. In these systems businesses collect and store large amounts of critical data that is sensitive and needs protection.

To mitigate attacks that could compromise the data on these systems, the system architects must design the system with security measures to ensure Confidentiality. Those measures include but are not limited to system segregation. Separate the data for the service need access to the data. Insure only people and application that have a true need to know have access to the data. The use of usernames and password, client and server certificates is a very important measure. The encryption and the compartmentalization of the data is also very important in ensuring the Confidentiality of the data. Separating the data based on the level of protection needed for the data is a key component to a comprehensive security plan.

The Integrity of the data is also a critical factor in a SOA security implementation. To mitigate the risk of problems with Integrity the system architects must design into the system security features that ensure the Integrity of the Web services. For example when a purchase is made online, the vendor will collect purchase card information and the send it to a third party for approval and payment processing. With the use of the loose coupling of the Web services, sensitive information is being passed from system to system. During these transactions the information is vulnerable to attack. As the information leaves one system and is routed to another system it is subject to interception corruption. The interception may come from simply eavesdropping on a unencrypted network traffic or through a more sophisticated man in the middle attack.

System Availability is the most critical factor when designing a SOA base system for business applications. If the system is not available for business then the business is not able to function. System down time could cost business millions and could affect customer confidence. If the customer is unable to interact with system, the customer could perceive the business is just as unreliable as their systems.

The biggest threat to system availability is from denial of service attacks. To help mitigate against denial of service attacks the system architects must plan for greater capacity than needed for the business. Not all denial of services attacks has malicious intent. The Colorado Rockies made the World Series and the company that sold the tickets online had not anticipated the demand for the tickets would be so large. At first it was thought the system was under a malicious attack, however it was later determined the system could just not handle the capacity.

One approach to defend against denial of service attacks is to apply smart traffic filtering. Keep critical system functions on a separate network away from public facing functions.

**Conclusion**

SOA is a driving force for future functional use within many organizations. However, these functions must be viewed as being shared services for all processes. Functional re-use will be the main means of ensuring that an organization can respond rapidly and effectively to market dynamics, and that improvements to specific functions will have the optimum impact across the whole organization. The key to having a secure SOA environment is to plan security for the beginning. Building a system and then attempting to apply security at the end of the development life cycle in ensure you system to be less secure than if security was part of the complete design process.

**Reference**

Erl, Thomas, Service-Oriented Architecture: Concepts, Technology, and Design Prentice Hall, August 02, 2005

**Michael Taylor,** Securing the Enterprise Service Bus: Protecting business critical web services, April 17, 2009