

Services Oriented Architectures: Managing the technical security risks and challenges

By Rafat Alvi

This article explains the evolution of Service Oriented Architectures (SOA), the new security challenges, and provides a holistic approach to solving key technical SOA security risks.

Over the past few years, enterprises have started adopting Service Oriented Architectures (SOA) to create faster, more agile business services. As enterprises embark upon a SOA, security challenges become a critical architectural issue. SOA exposes crucial business data to customers, business partners and, if not safeguarded properly, attackers. Compliance and regulations further complicate sharing of information with partners. This article explains the evolution of a SOA, the new security challenges and provides a holistic approach to solving key technical SOA security risks.

Architectural styles

The IT industry has gone through major architectural styles or movements. The client-server architectures led to the distributed three-tier architectures and then to Enterprise Application Integration (EAI), Enterprise Data Integration (EDI) and Business-to-Business (B2B) movements (see sidebar for details). With all these architectural styles and movements, enterprises often find themselves in a land of complexity – complexity of architectures, applications and IT Silos. Since, “the worst enemy of security is complexity,”¹ enterprises found themselves in an accidental architectural style, with little or no security.

Service oriented architectures solve the complexity of these architectural silos by wrapping the older architectural styles and encapsulating their core business capabilities as services. These services are then orchestrated into business collaboration across the silos, providing an end business process to

Pre-SOA architectural styles

SOA re-uses ideas from all the previous architectural styles. Client Server Architecture, the very first application architecture style, introduced a shared data resource (the server) processing multiple requests from clients. In SOA, each service implementation follows a client-server approach. Early implementations of client-server resulted in the mainframe: a single server side tier with thin clients with no application logic. This evolved to “fat clients” where desktop clients extended the mainframe logic. Fat clients quickly became costly to update and the distributed three-tier architecture was developed: fat clients were split into thin clients and a server side presentation and business tier. Earlier implementations of this style resulted in Common Object Request Broker Architectures (CORBA) and web-based standards like Java 2 Enterprise Edition. SOA assumes the presence of these styles and integrates well with their presentation tier. The downside of three-tier architectures was the cost of construction: enterprises found themselves building similar applications everywhere. Companies thus started selling ERP, CRM, financial, billing and vertical packaged applications to enterprise. As the complexity of these applications and their silos increased, enterprises found themselves needing to minimally connect these systems. The Enterprise Application Integration (EAI) and Enterprise Data Integration (EDI) movements were born. Enterprises also found themselves in need of making connections with partners and suppliers. A set of products that established such connections started being referred to as business-to-business (B2B) products. SOA provides a layer above all these architectural styles, unifying their access and integration.

1 [Http://www.schneier.com/essay-018.html](http://www.schneier.com/essay-018.html)

customers, partners and end users. Business owners decide the new “services” they want to offer, identify the business processes, and SOA allows IT ways to implement these business processes. The wrap and re-use model, if done correctly, allows for re-usable services to be exposed out of the existing architectural styles. Continuing customization of old architectural silos is then reduced, thus reducing complexity and increasing business agility.

Security challenges

In shifting the focus to re-usable services that wrap existing applications, SOA simplifies a lot of the current complexity. However it introduces some new security challenges that did not exist prior to SOA.

- **Business Services are now major enterprise entities and can be clients to other services:** Prior to SOA most architectural styles assumed users as calling principals. Authentication and authorization were done for the users only. In a SOA, services can act on behalf of users, organizations and departments. They have to be correctly authenticated, authorized and audited, above and beyond any users. Services can invoke other services as well, prompting the need to establish a trust chain across multiple service end points.
- **Transport level security is not applicable:** In the Pre-SOA days, message confidentiality, integrity and non repudiation requirements were implemented mostly by securing the transport layer by using SSL/IPSEC. This worked well in enforcing point-to-point security. However in a multiparty SOA collaboration over an inherently insecure network (like the Internet), this model breaks down. One can never be sure where the message might stop in the “cloud” and what intermediary might be receiving it before it is sent to its final destination. Regulations might dictate that only certain message parts are visible to only certain end points (social security to HR services, bank records to billing services). SOA needs to allow for message integrity and confidentiality across insecure networks and also for parts of message to be visible to certain parties.
- **Security controls in individual application services are not manageable:** Before SOA, each application embedded security controls into its business logic. Authentication and authorization mechanisms, keys and encryption strengths were application dependent. This model breaks down for tens or hundreds of services and no enterprise can manage changes to these security controls. SOA security architecture needs to allow for such management of security controls.
- **Security policies and procedures across trust domains are not standardized:** Prior to SOA enterprises established point-to-point connections across applications using non-standard protocols. The cost

of operating and managing a multiprotocol network kept increasing. A SOA needs to promote and integrate standard protocols and security policies within and across domains.

- **Traditional attack vectors take new forms:** SOA introduces new threats and vulnerabilities such as XML DOS, XML injection and XQUERY attack. Additionally service containers expose configuration files, passwords and database accesses to attackers who can leverage them to get to back end valuable data.

A holistic approach to SOA security

A holistic SOA security architecture needs to accommodate for all the above security challenges. It must:

- Provide a standard-based approach for providing for all security assurances to business services
- Allow for centralized management of security across various services
- Ensure message security in SOA interactions across trust domains
- Provide a secure infrastructure that mitigates internal and external attacks

We will now look at architectural approaches that fulfill these requirements

Securing SOA services

As SOA services grow in an enterprise, managing embedded security code per service becomes a tremendous challenge. A shared security service or component extracts security code out of application logic into a self-contained security service or component. Each security service (or component) manages a unique security assurance that is available for the business services to use. The following describes these key SOA security services.

An operational advantage for a separate security service or component is that they can be managed by a security team. This reduces the risk of an ill-informed or malicious developer creating a security policy that makes services vulnerable to attack.

User identity management service

Even though SOAs live in a world of services, the user is still the primary invoker of the collaborations. In a large enterprise, user identity is spread across multiple data stores. As users change roles, enter or leave the organization, their authentication credentials, passwords and roles change. A central identity management service provisions the users across the enterprise, updates the various identity repositories and ensures that SOA services are invoked by trusted identities. The identity management service can also be called by SOA business processes so that SOA services can provision and manage identities on demand.

Service provisioning services

Much like identity management, a service's life cycle also needs to be managed. Services have to be registered, provisioned (with appropriate credentials and tokens), version-controlled and deprecated. A central service provisioning and management service manages the life cycle of services, attaches security tokens (such as certificates) to services, renews expired tokens and deprecates tokens as services go out of commission. This service usually integrates with a service registry/repository where SOA services are housed and exposed.

Authentication and trusted token services

It is already common practice that enterprises have a centralized user authentication and authorization service. However many of the single sign on tokens provided by these services are non-standard. A Trusted Token Service issues standard-based tokens (for example user/password tokens, SAML assertions, X509 or Kerberos tickets). A trusted token service also brokers tokens, exchanges tokens for two different security domains, thus allowing interoperability and communication between disparate domains. Together the authentication and token services form the basis of authentication of services within a domain and across domains.

Authorization/policy service

In client-server days, authentication and authorization were combined in the same application and were coarse-grained, usually restricting access based on user role or group membership. However for a SOA service, much more fine-grained access control is needed, such as a policy decision based on service invoker, origination location, invocation chain, time of day, service requested, operation requested and any other conditions. A centralized policy service acts as an authorization authority to requests that have presented appropriate authentication tokens to a service provider and now require authorization. The policy service works in conjunction with policy enforcement point (acting as a security proxy) and applies fine-grained policy decisions to the incoming request, thus granting or denying access.

Message signing/encryption service

Message confidentiality and integrity are typically provided by signing and encrypting messages appropriately. It is not desirable for each SOA service to have to build its own signing and encrypting mechanisms and try to store its own keys and certificates – which allows for a wider attack surface. A common message signature and encryption service (or component) provides the appropriate signature and encryption algorithms per domain, uses the appropriate keys from the appropriate key stores to sign/encrypt the message and decrypts/verifies signature as needed. By housing these keys in secure stores, these services can further protect the keys. And lastly, since signing and encrypting large documents is per-

formance intensive, these services can offload processing to specialized XML processing hardware.

Federation services

Enterprises have legal agreements with customers to protect their identity information. In a SOA, different organizations (or parts of an organization) start to interact with one another to provide a value chain for the customer. In doing so, organizations need to identify the private attributes of their customers so that business partners can clearly provide business services. The challenge is that in the real world, regulations and privacy requirements restrict the passing of identity information without user consent. Federated identity services (such as Liberty SIS) provide for such trust environment between identity providers and service providers and allow for attributes that are consented by user to be shared. The user dictates which service providers see what, thus enabling privacy of information while keeping it shareable in a standard way.

Logging, auditing and monitoring services

Logging is done in many non-standard ways across application architectures. A centralized SOA logging service allows services to log their events in a standard way and store them in a tamper proof log store. Secondly, the auditing service is a separate service that audits events across logs stores and can perform complex rules-based auditing. The logging service can also interact with a monitoring service that proactively monitors the SOA security policy thresholds and notifies administrators about any violations.

Securing the runtime collaboration

Once the security code has been factored out of the application into centralized services or components, there are two architectural patterns that can be applied to invoke these security controls: a gateway or a co-processor.

A *security gateway* is a front and back controller to a SOA security domain. It intercepts all inbound requests to all services and outbound responses from all services and makes sure that messages follow defined authentication, authorization, encryption and signature policies. It can either provide common security components itself (such as encryption and signature) or collaborate with central services. Lastly, a gateway can mitigate new security threats (such as XML injection, XML Denial of service, XQUERY attacks, etc.) by filtering out requests for malformed or malicious content before they are processed by the business layer.

A security gateway often resides in a demilitarized zone (DMZ) but this makes it difficult if internal services can not access it. A *security co-processor* is a logical entity that can provide the same security services and components, in a peer deployment fashion to the business services layer. The co-processor provides a security layer to the architecture, receives requests for the security services and responds with the appropriate responses. It can also be realized with a combina-

tion of software modules (such as offering federation services) and hardware modules (such as ones doing encryption).

Securing the deployment

SOA services live in operating environments such as OS containers. Even though it might seem odd for SOA professionals to consider operating system and network security, it is almost imperative for them to do so. No matter how secure we make our messages, service containers can get hacked into. Attackers that get to service configuration files can misconfigure services, expose schema and access back end passwords. They can hijack ports, connect to databases, launch denial of service attacks and wreck havoc not on one machine but potentially on the entire SOA infrastructure. A *secure execution container* provides the following capabilities:

- Protects itself from unauthorized access or use by the services running within it
- Protects any service running within the container from unauthorized external influence
- Protects the IT environment (including service configurations, etc.) outside of the container (should a running service be compromised)²

These service containers could provide “levels” of secure configuration with a base configuration for all business services

² <http://www.sun.com/blueprints/0206/819-5605.pdf>

while a highly secure configuration (with trusted platform modules, trusted operating systems, key stores and fine-grained RBAC) for shared security services and gateways.

Conclusion

The security services, the security gateways, secure co-processors and the secure containers provide the necessary building blocks for a highly secure Services Oriented Architecture. As with any security architecture, these building blocks allow an evolutionary path to SOA security maturity. Coupled with good enterprise and SOA governance policies and procedures, this architecture allows enterprises to confidently expose new and innovative services to their constituents without fear of compromise and technical risk, thereby increasing their SOA agility and enabling IT to solve business problems faster, cheaper and safely.

About the Author

Rafat Alvi is a Senior Security Architect in Sun's CTO Office / Security Program Office where he leads the sales and engineering teams at Sun in creating repeatable security solutions. He has over 15 years of IT experience with focus on information security, web services, services oriented architecture and identity management. He is a frequent speaker at industry conferences such as Java One and RSA, is member of TRUST - security research consortia and has represented Sun in CORAL, a DRM interoperability alliance. He can be reached at rafat.alvi@sun.com