Systems Engineering at MITRE

SERVICE-ORIENTED ARCHITECTURE (SOA) SERIES

Information Assurance for SOA

J. J. Brennan, Editor

MITRE

Executive Summary

This paper addresses securing information technology (IT) systems having Service-Oriented Architecture (SOA) designs. The paper describes the challenges of securing SOA-based systems, discusses various security-related design alternatives for them, and, where practical to do so, provides specific recommendations on how to overcome these challenges.

An SOA-based system is an alternative to prevailing IT system designs that delivers functionality through loosely coupled and independent components, in contrast to the tight integration found in most existing systems. Although the security objectives for SOA-based systems—confidentiality, integrity, access control, accountability, and availability—are in almost all respects the same as those for non-SOA designs, securing SOA-based systems presents some unique challenges.

For example, SOA-based systems naturally support sharing information and capabilities across organizational boundaries in keeping with the stated goal of increased information sharing across the Federal Government. However, sharing and security are often in conflict and must achieve a proper balance. In addition, service use and delivery across organizational boundaries complicate the development of security requirements and responsibilities.

SOA-based systems often involve resolving tradeoffs, for example, deciding where to place security services or which services must authenticate to other services or service consumers while meeting accessibility and performance objectives. SOA-based systems can be incompatible with existing certification and accreditation (C&A) processes and procedures because they can be deployed incrementally, have difficult-to-define boundaries, can operate across organizations, and might support user populations that cannot be defined *a priori*.

This paper covers these challenges and how to meet them in five sections:

SOA Security Architecture discusses how SOA-based systems can deliver many security functions as services. It also examines services' message protection and service location, each with respect to possible system performance impacts.

Mediating Access to Services presents examples of how SOA implementations might use identification and authorization services as well as an example showing how chained service invocations can arise. It also examines alternatives for identity delegation and public key infrastructure (PKI) certificate use. Mutual authentication of security services and attribute-based access control also receive treatment.

Trust and Policy discusses the major concerns arising when SOA systems cross organizational boundaries, including system governance and security controls.

Audit and SOA-Based Systems deals with the facts that service consumer and provider interactions are often unpredictable, that services can be made discoverable, and that service provision crosses organizational boundaries, all of which make the task of building audit trails difficult. It discusses how orchestrating usage patterns can make SOA auditing feasible and presents options regarding audit record storage.

Certification and Accreditation for an SOA-Based System presents three tactics to cope with the challenge of obtaining C&A for an SOA system. The section also discusses how to describe services in terms of their commitments, provisions, and obligations as an aid to obtaining accreditation.

The discussion of each subject covers how delivering security differs with service vs. traditional architectures, provides illustrative examples, and summarizes key observations. The paper's intended audience is technology executives, system and security architects, and program managers who have an interest in securing SOA-based systems.

For more information on SOA, see http://www.mitre.org/soa.

Table of Contents

THE BIG PICTURE: Although the security objectives for SOA-based systems—confidentiality, integrity, access control, accountability, and availability—are in almost all respects the same as those for non-SOA designs, securing SOA-based systems presents some unique challenges.

Information Assurance for SOA

J. J. Brennan, Editor

The SOA Security Challenge

This paper describes the security challenges that accompany SOA deployments and discusses various ways to meet these challenges. Its intended audience is Federal Government executives and managers whose responsibilities touch the implementation, operation, or use of SOA-based systems.

An SOA is an alternative to traditional system design for delivering IT. An SOA delivers functionality via service components that service consumers, people, processes, other services, etc. reach through defined service interfaces. Service components are "loosely coupled," which is IT vernacular for saying that SOA components are minimally dependent on each other, an attribute that brings SOA implementations considerable flexibility. Thanks to this loose coupling and a reliance on interface specifications that allow consumers to obtain services, service providers can develop and test components incrementally and can also reuse available services for new applications or in ways not originally envisioned. Service consumers do not need to understand or deal with the details of service implementations other than the service interface. The use of metadata about services, made widely available in SOA implementations by placing it in service registries, makes finding and obtaining services both feasible and relatively easy. SOA-based systems naturally support sharing information and capabilities across organizational boundaries, even outside traditional enterprise boundaries, in keeping with the stated goal of increased information sharing across the Federal Government. Readers desiring an extensive discussion of the SOA approach to system design and the potential value obtainable from implementing an SOA are referred to a comprehensive discussion of both topics available in another paper in this series.[1]

Why Is Securing SOA-Based Systems Different

Securing any IT system can be challenging, and securing an SOA-based one is no less so. In addition to the usual security concerns—obtaining confidentiality, integrity, control of access, accountability, and availability—SOA-based systems pose a few unique challenges:

- Balancing information sharing with security
- Establishing trust across organizational boundaries
- Reducing vulnerabilities due the fact that some applications and services are exposed to networks
- Achieving certification and accreditation
- Authenticating and granting access to unanticipated users when the SOA implementation supports them

Balancing security with the need to share— A key benefit of SOA-based systems is the ease with which disparate organizations can communicate and collaborate. Information sharing is a primary

motivation for moving to a services approach for delivering information technology. The challenge here is to make services widely available to the broadest possible population of legitimate users while not compromising security.

Establishing trust—In contrast to traditional systems architectures, in which an owner organization has complete control of trust relationships, SOA-based systems require that these relationships be established and maintained across organizational boundaries to obtain the full benefit of SOA deployments. Establishing trust across organizational boundaries is often problematic as evidenced by how frequently the term "stovepipe" is used when describing systems.

Reducing vulnerability—All distributed computing raises concerns about possible vulnerabilities because access to system capabilities is granted to users and other systems dispersed across networks—SOA-based systems are no exception. Service interfaces, although able to be tightly defined, require proper input validation. Service particulars, including information about the service provider or technical details on how to obtain the service, are often published electronically. For some deployments, such information must be carefully examined to determine its sensitivity. Services also rely heavily on messaging protocols so that protecting information in messages while maintaining adequate performance can be challenging.

Certification and accreditation—The C&A process was developed when systems had few of the characteristics now found in SOA-based systems. For example, system owners deployed traditional systems with upgrades typically spaced by a year or more, not incrementally, with increments as short as days or minutes, as is possible with SOA-based systems. With traditional systems, owners can easily identify systems boundaries, in contrast to SOA-based systems where a system boundary is often effectively a network of connected services. Also with traditional systems, owners can control all configurations as well as interactions with external users. There is no need to deal with unanticipated users or discovery mechanisms because they do not exist. Thus SOA program managers are faced with the challenge of achieving certification and justifying accreditation using a process built for traditional systems with a set of characteristics that often do not match those of an SOA-based system.

Unanticipated users—With traditional systems, identifying all classes of users to be granted system access is part of system design. Identifying a system's user classes is necessary to establish the system's usage patterns, which helps in establishing the security controls needed and in ensuring that the controls will be in place. In contrast, an SOA precept is to allow users to dynamically discover and use services and information about them. Many SOA-based systems need to be able to authenticate and authorize legitimate users whose eventual use of services might not have been anticipated—so-called unanticipated users.

Meeting the SOA Security Challenge

Information and security officers, system and security architects, and program managers need to understand the SOA security challenge, including important design alternatives. The sections of this paper cover aspects of information security relevant to SOA designs with some emphasis on how security functions can themselves be delivered as services within a service architecture. The discussions cover how the topic differs with service vs. traditional architectures, provide illustrative examples, and summarize key observations

The sections on security architecture, mediating access to services, and security auditing focus on the different ways security objectives can be met. Because so much of how security is treated depends on the particular computing and threat environment and an organization's approach to and tolerance for risk, these sections offer little "do it this way" advice. However, although avoiding specific recommendations for these topics, the paper does illuminate the trade-offs involved and offers advice on when a particular course of action is appropriate. In contrast, the topics of trust and policy, and certification and accreditation lend themselves to prescriptions, some of which are given.

With a few exceptions, large-scale Federal SOA implementations have yet to emerge, and much of what is taking place in SOA practice across the Federal Government is still in the design, prototype, or small implementation stage. SOA security "best practices" are in a formative stage.

Readers should understand that what this paper describes as important to SOA security, including trade-offs, design options, and problem solution

approaches, represents the current state of the authors' knowledge. As noted earlier, SOA security objectives are similar to what would be encountered in any large, distributed system. The authors' many years of experience dealing with information security matters across a wide-variety of systems along with their understanding of various security standards provide the basis for our recommendations.

SOA Security Architecture

This section addresses topics that concern system and security architects as they consider possible high-level design options for securing an SOA-based system. It discusses how security functions can be delivered as service components of an SOA deployment and describes commonly used security services. It then discusses the role individual services play in delivering or consuming security functions, and how a service's security capabilities might affect a security design. Data assurance options are then treated, followed by a brief treatment of SOA boundary protection. Finally, that the location of security services relative to consumers and the use of cryptography can both have important performance implications are examined.

Why Is SOA Security Architecture Different?

When building security into SOA deployments, architects, engineers, and program managers should be aware of how to capitalize on a service architecture to obtain the benefits of an SOA approach as well as how to avoid potential pitfalls. Among the differences between securing an SOA-based system and securing traditional systems are:

- Security functions can be delivered as services within a service-oriented architecture.
- Different components have varying degrees of security capabilities, a fact that needs to be considered when deploying security controls.
- Traditional network boundaries are often porous to SOA messaging, particularly eXtensible Markup Language (XML)-encoded messages. Message-level inspection is often a requirement.
- SOA standards support confidentiality and integrity protection at the level of or within individual messages. Although flexible, providing message-level protection might degrade performance. Security architects need to carefully consider which of two approaches, message level or transport level, they will use to provide confidentiality and integrity protections due to possible performance implications.
- The location of security services needs consideration because where services reside affects both performance and availability.

Security Infrastructure Services

Security as a service—Because many enterprise systems require similar security functionality, it often makes sense to provide security functions as enterprise security services usable by any consumer needing them. Cser defines security services as "a set of reusable, standardized services (typically SOA-based) that provide applications with access management, entitlement, provisioning, attribute, auditing, and policy management products and services."[2]

Building an infrastructure with security delivered as services relieves individual system owners of the need to develop their own version of these critical services, which arguably improves consistency across the enterprise while decreasing errors and omissions. Another obvious benefit is economic—why build the same function *n* times when once will suffice? On the other side of the benefits ledger is the need for redundant and fault tolerant service designs to counteract the possibility that a single, critical security service is not available when needed.

Commonly implemented security service functions include:

- **Security policy decision:** Makes decisions on whether subjects, humans or systems,[3] have presented sufficient evidence to authenticate their identity or whether a subject should be granted a specific system privilege or access. Authentication and authorization policy decisions are often made by separate services.
- **Security policy administration:** Provides interface and functions for policy maintenance.
- **Security policy retrieval:** Fetches security policy from policy stores on demand.
- **Attribute retrieval:** Retrieves attributes about subjects, either human or systems.
- **Attribute administration:** Provides interface and functions for attribute maintenance.
- **Digital certificate validation:** Determines whether a PKI digital certificate has been revoked by the issuing agent.

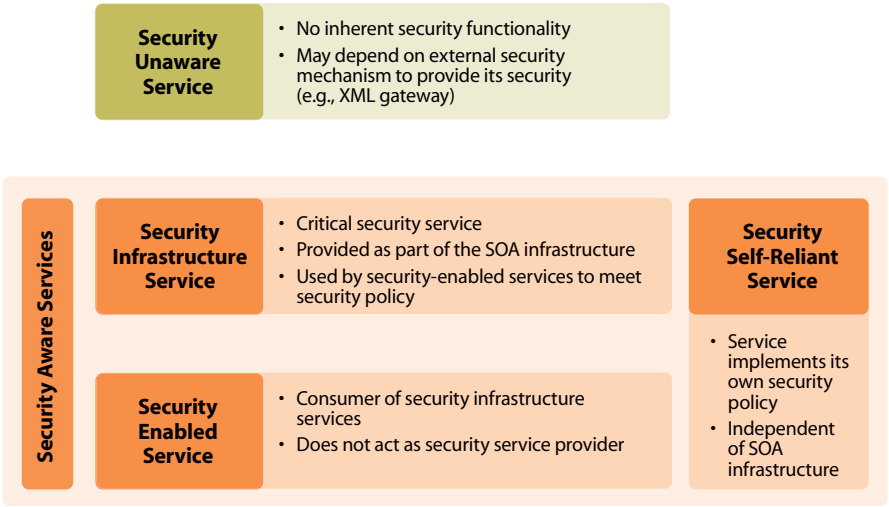

Figure 1. Security Service Categories

- **Audit storage and retrieval:** Accepts audit records and delivers them to storage; retrieves audit records when requested.

Although not unique to SOA deployments, policy enforcement needs mentioning. The locations where policy enforcement takes place, policy enforcement points (PEPs), are important in SOA deployments because PEPs are the places where the functions provided by other services such as authentication, certificate validation, attribute and policy retrieval converge to govern access granted to system resources or data. Given their role, PEPs are typically consumers (but not providers) of the security services available through the service architecture.

Security service awareness—It is beneficial to examine SOA services based on whether they consume or deliver security services and which type of service is consumed or delivered. This simple analysis helps illuminate those critical services that might need special provisions to maintain availability, as well as to guide the selection of the types and locations of standard security controls. Figure 1 illustrates security categories for SOA-based services.

Security Unaware services do not possess security features. They might receive some security services from other entities (e.g., various gateway and firewall devices) but are unaware that these services are being provided on their behalf.

Security Aware services require security, may take security relevant actions (e.g., permitting a user to access a resource), and are further delineated as:

- **Security Infrastructure services**—are part of the SOA infrastructure and provide one or more, typically critical, security services.
- **Security Enabled services**—primarily serve some business or mission purpose and use security infrastructure services as needed but do not provide such services.
- **Security Self-Reliant services**—implement their own security controls independent of an SOA infrastructure.

Table 1 presents examples[4] of the services shown in Figure 1. Note that security enabled and security self-reliant services typically act as PEPs.

Table 1. Service Category Examples

Service Category	Service	Explanation for Categorization
Security unaware service	Weather service	Responds to any subject with network access; relies on network security.
Security infrastructure service	Certificate validation service	Validates PKI certificates in response to requests from I&A services; part of SOA infrastructure.
Security enabled service	Flight plan service	Provides services to users per enterprise security policy; consumer of SOA infrastructure services.
Security self-reliant service	Legacy application	Provides security functions for itself but not for other entities.

Boundaries in SOA-based systems—System architects often define boundaries that group servers or services that have common characteristics or operate under similar assumptions. Boundaries

are the natural locations to place functionality that affects all the entities within them, with security protections being prime examples of functions that are often best located on boundaries.

As noted in the discussion of *The SOA Security Challenge*, service reuse is one of the benefits of an SOA implementation. Enterprises can make services available to components on a network, as a part of a service chain, or as part of an orchestrated series of service calls. However, because service location is often unrestricted, boundaries in SOA-based systems are not as distinct or as easy to identify as in traditional systems, usually making certification and accreditation more difficult to obtain (as will be discussed in *Certification and Accreditation for an SOA-Based System*).

Many SOA designs use HyperText Transport Protocol (HTTP) as a base transport mechanism, which requires that standard boundary security devices, such as gateways, open HTTP ports to traffic. Thus HTTP traffic, usually carrying XML-encoded data, flows through gateways—the traditional boundary protection devices.

Boundary protection—To the extent that services are located within network boundaries, XML gateways can provide these services with various types of protection. XML gateways are multi-purpose components, often packaged as hardware appliances running specialized software, which are in wide use in SOA deployments. Positioned as boundary devices, they can protect services within the boundary's interior by:

- Scanning for malicious software expressed within XML.
- Validating XML schema compliance.
- Validating WS-Security[5] clauses, including signature validation.
- Acting as a policy decision point (PDP) and a PEP by evaluating and enforcing an access control policy for each message based on its content and intended destination.

Figure 2 shows the position of an XML gateway in a typical deployment. XML gateways can scan XML content only after cryptographic tunnels such as provided by virtual private networks (VPNs) or transport layer security (TLS)[6]/secure sockets layer (SSL)[7] have been removed. The black octagons show where the terminus for a cryptographic tunnel usually resides, either on the network gateway itself or on some dedicated cryptographic device inboard of the network gateway.

Data Assurance

One of the benefits of SOA-based systems noted in *The SOA Security Challenge* is how well they support information sharing to organizations and users who would not have access to these services in traditional system designs. Wide sharing of services requires that service providers deliver data to consumers across networks that are out of their normal span of

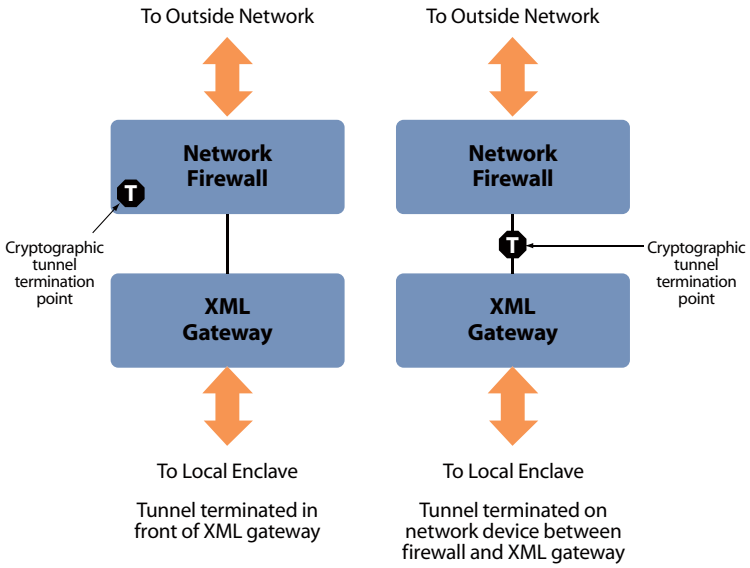

Figure 2. Typical XML Gateway Deployments

control. In most cases, this data requires confidentiality protection to ensure that only the provider and consumer have access to it. It also requires integrity protection so that the recipient can verify that the data received is the data that was sent.

***Confidentiality and integrity*—**Two general approaches, each with its own set of benefits and shortcomings, provide the required confidentiality and integrity:

- **Transport-level protection**—security mechanisms used to protect data at the network or transport layers of the communications stack. Besides providing confidentiality and integrity protections, most mechanisms support mutual authentication between the termination points of the protection. In practice, the termination points often are intermediate devices or systems, not the service provider or consumer. Examples of transport-level methods are Internet Protocol Security (IPsec)[8] (which is often used between network gateway devices), TLS, and SSL.

  Transport-level approaches create encrypted tunnels between the points where encryption is applied and removed. If some intermediate service or mechanism lying between the endpoints needs access to the contents of the tunnel, it is unavailable unless the intermediary is promoted to endpoint status.

- **Message-level protection**—security mechanisms that apply confidentiality protection, integrity assurance, or digital signatures on individual messages. Because protections are applied to messages, they extend from sender to receiver, achieving true end-to-end security when it is necessary. It is also possible to protect parts of a message so that only select recipients have access to the message's contents. Thus message-level protection is considerably more flexible than transport-level mechanisms. In addition, several standards exist or are in development to support message-level protection, for example, XML Signature.[9]

***Monitoring*—**Many organizations aggressively monitor activity on their networks by inspecting packet headers or packet payloads. The ability to monitor network activity clearly conflicts with the goal of providing confidentiality protections via encryption. Monitoring devices can only inspect unencrypted data, whose availability depends on the position of the monitoring device relative to encryption endpoints when transport-level protection is used. If monitoring is required, the monitoring device must be located between the encryption tunnel endpoint and the eventual receiver of the network traffic.

Using message-level techniques, portions of messages can be left unencrypted and thus transparent to monitoring devices. The important network information and all but the most sensitive message data can be made visible to network devices. It is also feasible to use different encryption keys for different portions of the message, which would allow selective viewing of message parts depending on having possession of the appropriate key. The drawback to all this is that providing protections on message parts is often more computationally burdensome than providing wholesale protection at the network level because PKI operations are used more frequently.

Performance Considerations

Security needs must be balanced with other considerations such as performance and scalability when designing SOA-based systems. To this point, Betancourt notes that "you must simultaneously maintain an approach that is risk-management based … you will see that some data must be more secure than others, some applications are more open than others, and so on. Therefore, don't treat all messages and applications the same; otherwise, there is an unnecessary security tax levied on the overall system that affects performance and overall system usability."[10] Smith affirms this: "When developing the security architecture for your SOA, remember that security always has an impact on the performance of service consumers and providers in your enterprise. Network calls made to authenticate subjects, retrieve authorization credentials, and obtain policy information have an effect on bandwidth as well as performance."[11]

Although performance is a concern in any distributed system, because SOA-based systems rely so heavily on underlying networks, security architects must consider how their implementations might degrade network connectivity or throughput. Additionally the location of services or their consumers relative to each other might make service delivery problematic due to network latency, outages, or other factors. Thus consideration should be given to whether services are provided by the enterprise or locally and under what conditions. In some SOA security architectures, service chaining, the calling

of services from within the implementation of other services, can create a surge of calls to security services such as an authentication service.

Locating security services—SOA architects should consider whether enterprise or local [12] services are appropriate in the SOA design, particularly when a mission-critical service may suffer a periodic lack of connectivity or when performance might suffer by too much reliance on enterprise services. For example, credential validation is a key security capability in an SOA implementation processing sensitive information—establishing and verifying the identity of a user requesting access to sensitive information is a critical first step in granting access. Such a system might implement its own credential validation service to authenticate the majority of users whose existence is known *a priori*, increasing performance and decreasing the chance of being unable to provide service. On the other hand, credential validation or attribute retrieval for unanticipated users might well be better implemented as enterprise services to relieve the local service of maintaining large directories of user information. As unanticipated user credentials are verified, credential information can be cached locally for a reasonable period of time to boost performance.

Table 2 gives suggestions for locating common security services and a rationale for choosing a location.

Table 2. Locating Common Security Services

Service	Local	Enterprise	Explanation for Categorization
Policy enforcement	X		Enforcement is typically required to be close (in a network sense) to the service being protected.
Policy decision	X		Being called for almost every message requires placement local to the calling service.
Attribute/policy administration/ retrieval	X	X	Enterprise-wide data is available and administered at the enterprise but is supplemented with local attributes and policies.
Certificate validation	X	X	Generally an enterprise service, but meeting performance needs may require a local cache or local service placement.

Two possible ways to manage connectivity to services are:

- **Enterprise, failover to local:** Use of enterprise security services is normal with locally run services used as a failover option when enterprise services are unavailable. Although this approach provides the greatest breadth of service and, in most cases, the best support for unanticipated users, it requires that systems know when the enterprise service is unavailable and then switch over to the local service. Latency is also a factor to consider when services are some distance (in terms of delivery time or hops) from service consumers on the network.
- **Local, with enterprise refresh:** Local service is the norm with periodic local data cache refreshment from enterprise data taking place when connectivity allows. The advantage of this arrangement is that local services can use enterprise data even when disconnected from the enterprise. The disadvantages are that local services risk making access control decisions with stale data, that enterprise data refreshment might require large data transfers, and that unanticipated users can only be supported if their authentication and access control data has been previously downloaded.

When the same service functions are implemented as local services in multiple locations, deploying multiple instances of a single, standardized service maximizes the benefits of using a service architecture.

Use of cryptography—Cryptographic operations are computationally intensive. Measured per byte processed, the public-key cryptography operations of decryption and digital signature are particularly demanding tasks, whereas their counterparts of encryption and signature verification can be implemented with less, although still significant, computation. Symmetric encryption methods such as the Advance Encryption Standard [13] (AES) are designed for high-speed encryption and decryption and are typically used for the bulk of encryption and decryption operations, with public-key cryptography reserved for session key delivery or digital signatures. Both IPsec and TLS (SSL) take this approach.

Many SOA standards support encryption and signature on all or parts of the messages passed between

SOA components. Computation to support message-level cryptographic operations can become intense and made worse because cryptographic operations might occur at many steps as a message is delivered across an SOA implementation.

Employing a policy such as "all messages must be signed and encrypted" is unlikely to deliver adequate performance using currently available technology, so such a policy might only be appropriate in highly sensitive environments. Conversely a policy of "no messages are signed or encrypted" might be perfectly reasonable when services are only used by a local enclave [14] that has significant boundary protections in place. The lesson is that SOA architects should be cognizant of the possible degradation in performance due to ill-advised use of cryptography and use it within context a system's threat profile and sensitivity.

Key Observations

1. The security controls used in SOA-based systems are generally similar to those used in prevalent systems. What is new with SOA-based systems is the ability to deliver many security functions as services within them. Security decision making and enforcement, security administration, and retrieval of subject attributes or credentials are prime candidates for implementation as security services.
2. In delivering security in an SOA-based system, it is useful to inventory the security capabilities of services and applications to help understanding what security controls should be in place as well as where they are needed. Understanding the security capabilities of the SOA components also helps in developing redundancy and availability plans.
3. Boundaries in an SOA-based system are amorphous. Traditional firewalls pass port 80 traffic used by many XML-based protocols. XML gateways are used to inspect message-level traffic for integrity and acceptability.
4. Message traffic between services often needs cryptographic protection. The usual choices are transport-level protection, implemented on the network, and message-level protection, implemented on all or parts of messages. Message-level protection offers considerable flexibility but must be used judiciously to avoid possible performance degradation.
5. Transport or message protection can deny network devices the ability to monitor network traffic.
6. The location of services is an important consideration due to possible network latencies and problems with connectivity.
7. The use of cryptography can create performance problems in certain cases. SOA designers should ensure that cryptographic use satisfies a real need.

Mediating Access to Services

In most cases, an SOA-based system employs identification and authentication (I&A) as well as access control functions prior to allowing human or machine access to its services. [15] Depending on the specifics of a particular service-oriented architecture deployment, SOA components can deliver, consume, or provide only for themselves either authentication or access control services. This section deals with how I&A and access control fit with the SOA approach to delivering IT functions, particularly how these two important security controls can themselves be embodied as services, and then addresses a few design considerations relevant to access.

Why Is Mediating Access in SOA-Based Systems Different?

Although the goals of authenticating users and determining whether to give them access to resources are independent of the type of system in operation, use of the SOA approach has been accompanied by procedural changes in how resource access is accomplished.

Use of access services—As noted in *SOA Security Architecture*, with an SOA design approach, security functions such as I&A and access control can be delivered as services.

Unanticipated users—In many SOA implementations, users can discover services as they need them. Ensuring that only legitimate users gain the access allowed by their set of privileges is challenging.

Chaining—SOA-based systems make chaining services possible. In the simplest case, a subject sends a service request to Service A, which fills the request by invoking another service, B. The possibility that service chains will arise can complicate both the development of security policy and the mechanics of access mediation.

Reliance on PKI—The use of PKI functionality in SOA-based systems arises for at least three reasons:

- Organizations such as the DoD are moving to PKI-based authentication for all users. As a result, SOA services in the DoD or similar environments must support PKI use for a large segment of their user populations.
- SOA-based systems link providers and consumers through a service interface. Having each service define its own authentication interface would likely make SOA-based systems unworkable—each consumer would have to comply with multiple, and likely different, authentication methods. Using a PKI helps solve this problem because all services can rely on a single, common method.
- Many SOA standards rely on public-key cryptography functions, for example, Security Assertion Markup Language (SAML) v2.0. [16]

Attribute-based access control (ABAC)—Allowing users to access resources as a function of their attributes (which could include their identity or role) and separately expressed policy rules appeals to organizations that want to share information widely. By using attributes, access can be granted dynamically based on the situation. Although ABAC is not unique to SOA-based systems, the SOA approach to delivery aligns well with how ABAC operates functionally.

Identification & Authentication

To help motivate the discussion of some considerations that matter when I&A is implemented as an SOA service, here are two examples:

SOA authentication example—Many service configurations for I&A within an SOA implementation are possible. Figure 3 depicts a simple, yet realistic, scenario. This example illustrates how authentication as a service might operate.

At Step 1, the user seeks access to an application, which happens to be a security-enabled application. Recall that the application is security enabled if it is a consumer, not provider, of security services. The application uses an enterprise authentication service to authenticate the user. It sends the user's identity in a request to authenticate the user to that service (Step 2). The authentication service might have several authentication methods from which to choose to authenticate the user (PKI, time-based hardware token, password, etc.), and it could actually choose whichever method is relevant based on context. It might also contact other services, for example, a service that validates a PKI certificate in fulfilling the authentication request. At Step 3, the authentication service requests information from the user. In the case of PKI-based authentication, Step 3 typically involves a challenge requiring a response. The authentication service then makes a decision whether to authenticate the user based on applicable policy retrieved from a policy database (Step 4). Finally, the authentication service returns its decision to the requesting application (Step 5).

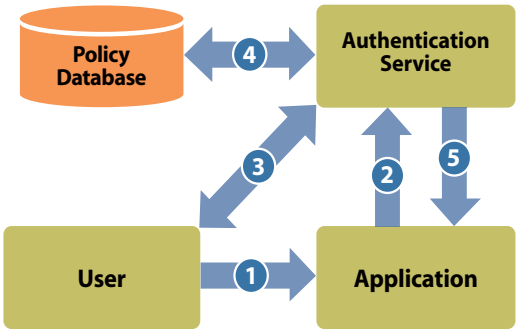

Figure 3. Authentication Service Example

Chaining example—The previous example shows how authentication might be implemented as a service or set of service components. This example illustrates how chained services and authentication interact and what security design decision the SOA security architects need to resolve.

Consider the situation in Figure 4 where a user's browser sends Service Request 1 to Service A, which then authenticates the user using an authentication service not depicted in the figure but similar to the previous example. Service A could then invoke an authorization service to determine what actions the user could pursue at A. However, if the requested service at A requires the use of Service B, shown as Service Request 2, B must ensure that it is dealing with a legitimate requestor, either the user or A. Service A could make the request on its own behalf or forward a request from the previously authenticated user, sending an assertion as to the user's identity with the request. Depending on the situation and governing policy, B might also check to ensure that the entity requesting access to the services or data requested is authorized to obtain access.

Figure 4. Service Chaining

Authorization Process

"An 'authorization' is a right or a permission that is granted to a system entity to access a system resource. An 'authorization process' is a procedure for granting such rights."[17] Just as with I&A, almost all systems must deal with granting access to resources as requested by subjects, each of which has a user, hardware, process, application, or service identity. The process of authorization involves:

- Delivery of an access request from a subject, possibly through an intermediary, to an access decision or enforcement process.
- Obtaining sufficient information about the subject to match the subject to an access policy. This might involve interacting with the subject directly or it might involve an intermediary. The subject's identity might not be needed, for example, if a trusted intermediary vouches that the subject belongs to a particular group whose access rights can be determined.
- Obtaining relevant policy for the subject-request pairing.
- Determining, based on the subject, request, and governing policy, if the subject should be granted access.
- Delivering this decision to the relevant access enforcement point.

Mutual authentication— The example depicted in Figure 5 illustrates one realization of an authorization process that relies on various services to complete. A subject requests service from a service component, which, in this example, acts as the authorization PEP. Where policy enforcement takes place can vary, so the example presents one possible alternative. The PEP asks another component, the PDP, to decide whether to allow access by the subject. In this example, access is granted based on subject attributes, which the PDP retrieves by using an attribute service. The PEP then asks a policy retrieval service to fetch the applicable policy for this subject's attributes and request. The PDP then decides if the subject should be granted access and returns this decision to the PEP.

Service-based authorizations can quickly become resource intensive when each component of an authorization chain is required to authenticate either the original requestor or some service down a chain of services acting to fulfill the original request.

Although the example uses attributes, authorization processes can be of various types, all of which could be supported by an SOA-based system:

- Identity (identity-based access control)
- Group affiliation (group-based access control)
- Role being performed (role-based access control)
- Attributes of subject (attribute-based access control)
- Attributes of subject, environment, session, and context (risk-adaptive access control)

SOA I&A and Access Control Design Considerations

The preceding examples suggest that I&A and authorization, implemented in a service architecture, can quickly become a complicated affair. Each service in a chain of services needs to ensure that the entities it is dealing with are legitimate and that in providing its service, it does not provide access rights to the requestor greater than those to which it is entitled. Several considerations arise:

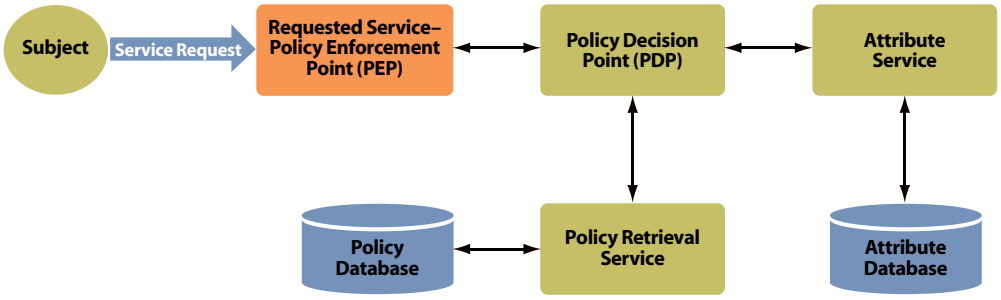

Figure 5. Authorization Example

- Should the subjects using an SOA-based system mutually authenticate, and if so, how?
- What is the policy guiding authentication as services are invoked in a chain?
- Should intermediate services require authentication of the user or just the invoking service?
- If the SOA implementation uses PKI certificates, which subjects should possess certificates?

Credential delegation—In the case where service fulfillment involves a requesting subject and a single service provider, there is no choice about which access rights and which identities to use. As service chains become longer, service providers along the chain can make I&A and authorization decisions based on the original subject's identity and access control privileges or on some intermediary service's identity and access control privileges.

With the first option, all service providers must authenticate the original requestor. Services could do this directly, perhaps by invoking an authentication service as described earlier, or by accepting an assertion of authentication (using SAML) from an intermediary service. There is some question as to whether adequate performance can be maintained if each service in a chain has to perform an authentication or check a SAML assertion when transaction volumes are high.

The other notable option is to operate under a delegation policy where the original requestor allows intermediary services to act on its behalf. Provider services then rely on I&A and authorizations for the requesting service rather than on those of the original requestor. This approach has the advantage of hiding the original requestor's identity and access rights from the provider for those cases where such precautions might be necessary. This approach can also considerably lessen the computational burden of authentication and authorization, depending on the SOA design approach to trust between services, which might differ from the approach to trust between a service and user.

This option is not without disadvantages. In some cases, governing policy requires that access be granted only with knowledge of the ultimate access consumer, which clearly is not supported in the delegation scenario described. Also, in the absence of assertions about the ultimate consumer made by intermediate services, the final service in a chain may not be able to make an access decision at all—service access might depend on attributes of the original requestor. Suffice it to say that whether or not delegation is appropriate or beneficial depends on factors like the operative trust model and security policy as well as operational considerations.

Mutual authentication—In an SOA-based system, requesters might authenticate the service they are dealing with so they can trust what they receive. For example, in Figure 4, the original requestor might authenticate Service A, either as a result of policy or caution. At the same time, providers need to know enough about a requestor, either the original requester or an intermediary, to determine access rights. The examples make this clear. However, it does not follow that all authentications require PKI-based authentication, SAML assertions, or messages with encrypted and digitally signed fields because a policy of "validating everything" can have serious negative performance consequences.

Alternatives to full-scale PKI-based mutual authentication work effectively in many practical situations. Many organizations operating servers within a tightly controlled enclave or computing environment implement a policy whereby all services or applications resident on the servers implicitly trust each other. Another alternative is the use of IPsec, which allows IPsec tunnel endpoints to mutually authenticate using strong methods at tunnel setup. A single network-level authentication applies to multiple service requests and provisions flowing through the tunnel. It is prudent to look at alternatives to a policy of validating identities for every service request. Except in the most restrictive environments, alternatives to full-scale mutual authentication might be preferable.

Certificates—If the services in the chaining example are using PKI-based authentication, Service B would use Service A's PKI certificate in authenticating A. An important practical question is whether services should have certificates at all (a burden to PKI administration if there are many services) or whether the physical server should have a certificate that is used by all services running on that server (effectively granting all services the same rights because they share a common authentication identity). Additionally, if a relying entity uses a server certificate to authenticate a resident service, the entity needs to establish that the service actually resides on the authenticated server. Typically, the relying service knows (or assumes)

that non-technical controls are in place, e.g., service agreements or server management policy and practices.

Access control policies—This paper frequently references policies governing I&A and authorization without saying much about where such policies come from. The use of security delivered within services, unanticipated users, ABAC, and other typical features of SOA-based systems has created the need for a standard approach for dealing with policies. One option for dealing with policies is to use the eXtensible Access Control Markup Language (XACML).[18] XACML, encoded in XML, is a language and platform-neutral method for representing and enforcing security policies.

Attribute-Based Access Control and the Unanticipated User

As noted earlier, one possible way to mediate subject access to resources is to use subject attributes in an attributed-based access control process. However, it is worth noting that *which* attributes to use with ABAC to properly control access is not obvious, and finding an adequate set of attributes might prove challenging. Not only do the chosen attributes have to define the set of subjects who will gain access and no others, but they also have to be accessible to services, available when they are needed, and accurate. It is entirely possible that the attributes that are actually accessible and available do not adequately define the set of subjects that should be granted access, a situation that precludes the use of ABAC.

Though ABAC use is not unique to SOA-based systems, it fits well with the SOA style of delivering security services and it supports dealing with unanticipated users. By implementing policies based on attributes instead of user identities or roles, enterprises can securely mediate access to services for users without prior knowledge of their existence or need. For example, a system could grant access to certain resources based on a user's attributes of parent organization and security clearance.

Implementation of an ABAC process needs to proceed with some caution. The following points need consideration:

- Access to the attribute data store must be mediated. For example, both the PDP and Attribute Service in Figure 5 must be authorized to access the Attribute Database.
- Highly detailed access policies might be needed for some attributes, for example, if the existence of certain subject/attribute combinations needs to be shielded.
- Detailed access policies might be needed to comply with privacy restrictions, for example, when user attribute sets contain personal information whose dissemination would violate privacy rules or regulations.
- Authorizations that require authorizations can lead to infinite recursion.
- Attributes might be misused, for example, by reusing an attribute assertion at a later time when the assertion might be taken for valid when the attributes behind it have changed. Having the attribute service timestamp and sign attribute assertions helps avoid misuse.
- Subjects might conceivably perform data mining on the store of attributes and subsequently misuse the information gained. Monitoring for data mining activity and placing limits on the number of responses from an attribute service might be required in some situations.

Key Observations

1. SOA implementations can include services that support I&A and authorization.
2. Although PKI usage and ABAC are not inherently part of an SOA-based system, their use fits conveniently with the SOA design approach.
3. The possibility of chained service calls to support I&A and authorization requires that system owners carefully examine various aspects of their SOA security design, including how delegation is affected, when and how mutual authentication is required, and what infrastructure pieces have certificates.
4. Having all service providers authenticate an entity on every service call can lead to undesirable performance. System owners should decide when and where strong authentication is needed. Alternatives, such as the assumption of trust based on location, or network-level authentication might be justifiable in lieu of an "authenticate everywhere and every time" policy.
5. ABAC fits well with the SOA-style of service delivery. Organizations contemplating ABAC use should assure themselves that the available attributes are adequate to the task envisioned.
6. Various standards (e.g., SAML and XACML) support security services implementations.

Trust and Policy

A key potential benefit of deploying SOA-based systems is gaining the ability to easily interact with users or services outside organizational boundaries. In contrast to traditional implementations, where a governing entity controls access to resources and establishes security policies, SOA implementations that cross organizational boundaries require that trust be established between the cooperating entities and that they agree on how services will be accessed and by whom. Baer notes: "Managing trust, and with it, entitlement to the capabilities of services, requires a far more granular strategy for authenticating the user, granting authorization, and providing access that depends, not only on the user's location or organizational affiliation, but also on the policies that apply to the particular Service." [19]

Why Are Trust and Security Policy in an SOA-Based System Different?

Agreeing to cross-boundary security policies— Traditional systems operate within boundaries set by system owners who also define usage and security policies. When SOA services span organizational boundaries, the organizations involved need to reach agreement on usage and security policies.

Accessing and complying with security policy— Users dealing with traditional systems usually have little trouble finding or understanding the security policies that govern their behavior—system owners provide this information for them. Users accessing SOA-based systems, especially when the service owner is not their parent organization, might have difficulty discovering the operative restrictions and procedures or complying with them. This difficulty might arise for a variety of reasons. The provider organization might not have planned well or made assumptions about users that are not true or are difficult for them to meet. The service owner organization might operate in ways foreign to service consumers. Difficulties might also appear if certain services are accessed infrequently or in an ad hoc manner, requiring users to adapt their behavior to changing services environments.

Establishing Trust Across Organizational Boundaries

Elements of trust— Organizations engaged in providing or using SOA services that cross organizational boundaries need to establish applicable rules of engagement. The rules of engagement include, among other things, how the organizations will trust each other and each other's members and what policies and procedures govern the use of the SOA services. Specifically, each of the following topics deserves treatment:

- Identification of services to be provided by each organization
- Enumeration of security mechanisms required for each service
- Agreement on trust to be afforded to users
- Characterization of unanticipated users (if supported)
- Governance
- Allocation of governance responsibilities

Service-level agreements (SLAs) and memoranda of understanding (MOUs) are typical ways to articulate the agreements made.

Security mechanisms and controls— Service agreements should specify security controls that each party commits to deploy and what obligations fall on the counterparty. [20] Examples of statements addressing security controls are:

- Consumers of Service X shall provide SAML assertions confirming attributes A1 and A2.
- All service invocations to Service X shall be digitally signed by the consumer.
- All service responses shall be encrypted using XML-Encryption.

Service agreements should also address the type of I&A and authorization controls applicable to classes of users as well as the circumstances under which different controls might apply.

Governance— Governance is "a decision and accountability framework to encourage desirable behavior." [21] When SOA implementations cross organizational boundaries, the organizations involved must come to an agreement about this "decision and accountability framework," with security and availability being important elements in such agreements. Governance agreements should specifically address:

- Monitoring—which characteristics of service agreements will be monitored for compliance.
- Monitoring party—which organization(s) has the responsibility for monitoring these characteristics.

- Non-compliance—what happens when non-compliance with service agreements arises, and, in particular, who is responsible for taking action in the event of non-compliance.

Lifecycle schedule—Whether dealing with an SOA-based system or not, addressing security early in the development process lowers both program risk and cost to service provider organizations. Governance policies and SLAs should be developed in parallel with the development of system or service requirements. These policies and agreements should be in place prior to the development of service implementations.

Defining and Implementing Security Policies

Security policy establishes a common understanding between organizations, and SLAs codify that understanding. Security policy should be written in consumer-readable, vendor-neutral format that allows a consumer to understand the security controls necessary to access a service.

Various standards exist to allow service providers a consistent way to publish their security policies for service consumers. Among these standards are:

- WS-SecurityPolicy[22] defines a framework for allowing web services to express their constraints and requirements as policy assertions in XML format.
- Web Services Description Language (WSDL)[23] uses an XML format to describe the public interface to a Web service and to define services as collections of network endpoints.
- Universal Description Discover and Integration (UDDI)[24] is an XML-based registry for services.

Key Observations

1. Organizations using SOA designs that cross organizational boundaries need to define rules of engagement that cover how security is provided, what trust is afforded to users, and how governance of the service architecture will take place.
2. Interoperating organizations should capture trust and policy agreements in SLAs and MOUs.
3. Security policies should be written in human-readable format, vendor-neutral format that allows consumers to understand the security controls necessary to access the service.
4. Organizations should establish security policy early in the development lifecycle as requirements are formulated.
5. Standards based on XML are available to help service providers publish their security policies for service consumer discovery.

Audit and SOA-Based Systems

Audit records are used to build a history of system usage, for example, of who accessed a system and at what time, what applications were invoked, or what data records were altered or deleted. Audit records have two major uses: user accountability and post-event forensics. Because many system users—government and commercial—deal with sensitive information and have, by necessity, considerable freedom of action when dealing with such information, organizations use audit trails to promote responsible user behavior. Management can review audit trails to monitor compliance with an organization's data and system access policies. When used in forensics, audit trails help investigators analyze events such as network intrusions, crimes, and user misdeeds. Fulfilling audit requirements is not just a matter of prudent operations as various Federal publications[25] contain audit requirements or guidelines.

Why Is Security Auditing in an SOA-Based System Different?

Audit trail creation is challenging regardless of the architecture in place or operational environment. With an SOA-based system, this difficulty is compounded because services, and the resulting patterns of service usage, may be distributed across system and organizational boundaries. Building a history of a single transaction can be a complex undertaking because:

- Usage patterns (e.g., the extent to which services or users dynamically discover SOA services) can make the creation of audit trails significantly more difficult than with traditional systems.
- Service delivery across organizational boundaries is more likely than with traditional systems, which requires more planning and effort to support audit trail creation.
- Unlike almost all other SOA functions, little in the way of standards exists to support SOA auditing.

At the same time, implementing a services approach to auditing can make collecting and analyzing audit trail events considerably easier than trying to track and analyze auditable events across myriad systems, each with their own collection and data format peculiarities.

SOA Usage Patterns

How enterprises enable various usage patterns affects how easily they can build audit trails. By way of comparison to an SOA-based system, consider security auditing in a monolithic system. With a monolithic or legacy system, it is usually straightforward to identify which user, software thread, or system process initiated an event. In addition, a system-owner-approved design specifies the format and content of audit records. However, in an SOA-based system, services may be discovered and used by subjects not anticipated when the services were deployed, may be composed into chunks of functionality on the fly, and may be operated by a variety of organizations in or across enterprises. In addition, enterprises can operate service architectures in various ways: via orchestration, choreography, or dynamic discovery. The choice of operating pattern makes security auditing more or less difficult.

Orchestration—Linthicum defines orchestration as "a standards-based mechanism that defines how web services work together, including business logic, sequencing, exception handling, process decomposition, including service and process reuse."[26] With orchestration, a governing entity directs the coordinated use of services to produce a services-enabled capability. The entity has substantial knowledge about how services are being stitched together to provide the capability. Due to the entity's knowledge of how services will actually be used, implementing audit mechanisms in an orchestrated architecture is relatively easier than with other usage patterns.

Choreography—With choreography, each service determines the next service(s) to be invoked to fulfill a service request. The result is that service fulfillment can create a chain of services, each of which has only a local view of the entire service transaction. To obtain a complete audit picture of the entire chain requires that individual audit components from each service be stitched together. Constructing a complete record may not be feasible, depending on such factors as service chain complexity and the agreements governing service use.

On the other hand, the existence of audit records within each constituent service may be entirely sufficient for forensic analysis purposes, for example, if the audit records can be correlated after the events being analyzed took place. However, with large-scale operations, where transaction volumes of 10–100 thousand transactions per second are not uncommon, creation of an audit trail might be problematic even if timestamps are used (the clock error between systems is greater than the time between events). The problem of creating audit records in situations with high transaction volumes is surmountable, at least in theory, by creating a unique transaction ID that passes from service to service. However, the use of transaction IDs raises the possibility of introducing dependencies between otherwise independent services.

Service dynamic discovery—With service dynamic discovery, each service is discovering other available services that fulfill its current needs and then invoking them to piece together a capability. Though this style of use represents SOA design in its purest form, given high transaction rates and the dynamism involved, achieving a complete audit trail may well be impossible.

In light of this discussion, if organizations implementing SOA-based systems face stringent audit requirements, the orchestrated usage pattern is the only one of the three with a reasonable hope of meeting such requirements. Except in rare cases, say, with very low transaction volumes, stringent audit requirements will not likely be met with either a choreographed or dynamic discovery usage pattern.

Location

Besides adapting audit record collection to service operating patterns, security architects must deal with the fact that audit records can be of value to both sides of a service arrangement crossing organizational boundaries. In addition, where records are stored affects their accessibility, hence utility.

Crossing organizational boundaries—When service delivery crosses organizational boundaries, collecting and distributing audit data requires some additional planning. Organizations must have agreements in place covering what events

will be audited, what formats will be used for audit records, and how audit records will be delivered. Cooperating organizations can use MOUs and SLAs to document their cross-boundary audit policy.

Centralized vs. local audit stores—In SOA-based systems, audit record storage can be local to the service provider, centralized at a location agreed on by service providers, or some hybrid of the two. The storage choice should consider the potential benefits of each option; the costs of hardware, software, management, and administration; the costs and other burdens involved with disposition of the audit records; and the sensitivity of the data contained in or possibly inferred from the records.

If the SOA implementation uses local stores, service provider organizations reap the benefits of being able to use their own formats and maintain control of the audit data. However, locally stored audit records pose a significant challenge to creating a complete audit trail for service fulfillment with multiple steps. Construction of an audit trail would likely have to deal with dissimilar audit record formats, time synchronization problems, and different ways of creating audit record ID numbers.

Centralized record storage has the potential to simplify audit trail construction, once matters of common record formats are addressed either by agreements among providers or by back-end processing. Centralized record storage has the shortcoming that the record database, taken in its entirety, may contain more, possibly highly sensitive, information than what examination of individual records would reveal.

Audit Record Processing

Unlike most SOA functions for which numerous standards (115 standards as of April 2007) [27] exist, no approved SOA audit standards are currently known to the authors. Unfortunately, there are no standard protocols or defined exchange formats to help with the task of correlating audit data. This deficiency might prove increasingly problematic as the Federal Government increases it use of service-based systems and architectures. The lack of audit standards can hamper an enterprise's ability to realize the potential benefits of "audit as a service" alluded to earlier, especially when audit trails cross organizational boundaries.

Another point to consider is that in forensics investigations, it is sensible to ask to what extent application-level audit records will be used. A cost-benefit analysis, even an informal one, is appropriate before committing to collecting and storing application-level records.

Key Observations

1. SOA designs implement security auditing for the same reasons that traditional systems do: accountability and forensics.
2. SOA-based systems can make the task of security auditing challenging for various reasons, among them are that service consumer and provider interaction patterns are often unpredictable, that services are discoverable, and that service provision crosses organizational boundaries.
3. The extent to which service usage patterns are orchestrated makes audit creation more or less difficult. The more orchestrated usage patterns are, the easier it is to create audit trails.
4. If an organization is faced with stringent audit requirements, an orchestrated usage pattern is the best, and possibly only feasible, choice.
5. Whether to store audit records locally or centrally depends on the situation. Local record storage offers convenience to local service owners in examining auditable events at the expense of making global audit trail creation more difficult. Central record keeping is more amenable to audit trail creation. However, central record databases may contain more information, derived by making inferences across individual service or application record trails, than is acceptable.
6. Little exists in the way of standards to help SOA auditing.

Certification and Accreditation for an SOA-Based System

The goal of C&A for a system with a service-oriented architecture does not differ from that of C&A for a conventional system: to describe how and to assess the extent to which the system meets security control requirements in its operational environment. SOA-based systems pose particular certification and accreditation challenges.

Required security controls and C&A processes vary, depending on whether the owner of the system being certified and accredited is a civilian agency or the military and whether it is a national security system (NSS).[28] This discussion uses the DoD (Department of Defense) Information Assurance Certification and Accreditation Process (DIACAP) as a specific example of how to deal with C&A. Similar concerns regarding C&A for SOA-based systems arise in other communities, and the approach suggested here is relevant in concept for non-DoD environments.

Why Is C&A of an SOA-Based System Different?

Service-oriented architectures exhibit certain characteristics that pose a particular challenge to the current C&A system:

- Services—particularly business or mission application services—in an SOA-based system may be more likely to be deployed incrementally and asynchronously evolved.
- System boundaries may be more difficult to define.
- External services are likely not under the system's configuration management control.
- Users may be unanticipated.

C&A for SOA-based systems—The phrase "C&A for SOA" can mean different things, depending on the scope of the certification and accreditation. The SOA architectural model (Figure 6) defines different subsystems or categories of system components, with different C&A scopes.

Referring to Figure 6, the SOA Infrastructure (including, for example, an Enterprise Service Bus or a Service Registry) runs on a foundation of hardware and software. The Hardware & Software Foundations can provide information assurance (IA) controls (e.g., partitioning or process isolation) to the SOA Infrastructure and to services in an SOA-based system. The SOA Infrastructure provides supporting functionality, but generally not IA controls, to SOA Infrastructure Services and SOA Application Services. SOA Infrastructure Services can provide a variety of IA controls to SOA Application Services, in particular identity and access management (IAM) and auditing. SOA Application Services can implement IA controls, for example, by restricting data access based on a user's history of prior accesses. Business transaction functionality enables agreement on policies and requirements.

So, what does "C&A for SOA" refer to? Most commonly this phrase is used in one of the following ways:

- C&A of a set of SOA Infrastructure Services that provide security controls.
- C&A of an integrated set of components that includes a set of SOA Infrastructure Services.
- C&A of an SOA Application Service.

It is also important to understand what "C&A for SOA" does not refer to: non-discoverable, web-based services in a single-enclave environment.

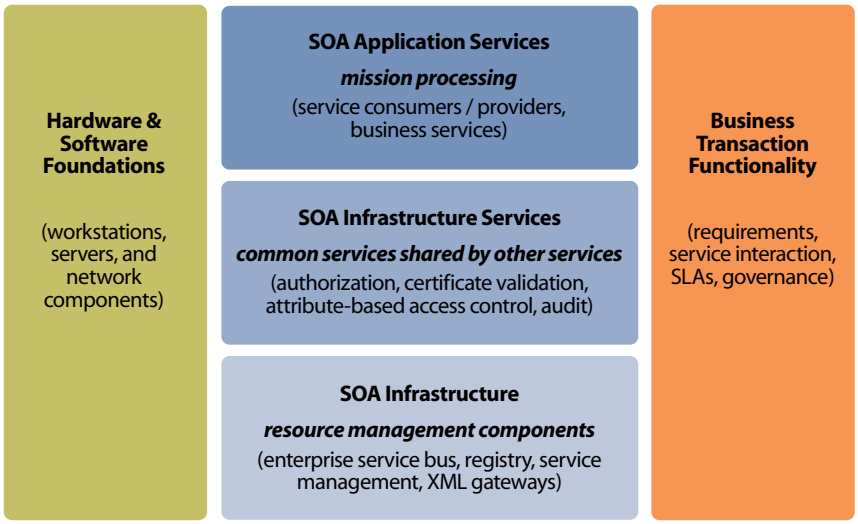

Figure 6. SOA Architectural Model

A Strategy for C&A of an SOA-Based System

The goal of C&A is to support the management of security risks. As noted earlier, service-oriented architectures present new challenges for identifying and assessing risks. In an SOA-based system, these functional components may be deployed at different times; for example, the underlying hardware and software foundation could be swapped out without changing the SOA infrastructure or services, or a new service could be deployed. The same SOA infrastructure service could be replicated in multiple enclaves. A change in the mission environment could cause a set of mission application services to be chained in a way not foreseen by existing mission or business rules. The following subsections describe how existing C&A processes can be applied to meet these challenges.

According to IBM, "Traditional IA C&A processes, although well suited to the traditional serialized 'waterfall' development model, do not support the continuous incremental SOA development cycle effectively. Further, SOA-based system complexity will require that IA personnel actively engage in analysis of system architecture and implementation much earlier in the lifecycle in order to keep pace. Executing C&A on a completed SOA-based NSS will prove infeasible if IA teams attempt to do it as a milestone activity."[29]

Approaches to meeting the SOA C&A challenge—To help them meet the C&A challenges, enterprises can use three approaches:

- Careful scoping of C&A
- Type accreditation
- Inheritance

Any of the major C&A processes—the NIACAP, the DIACAP, the IC process,[30] and the process defined by NIST 800-37—allow the scope of C&A to be defined so as to facilitate information assurance risk management for the enterprise. By scoping C&A carefully, different packages of system components—hardware and software, the SOA infrastructure, an individual SOA infrastructure service, an integrated set of SOA infrastructure services, or a set of application services—can be certified and accredited separately.

Type accreditation enables a package of components to be certified under a set of defined assumptions about the technical, operational, and threat environments in which it will be used. With inheritance, one system (or package of components) can implement an IA control and provide that control to another system, which thereby "inherits" the control. Taking advantage of type accreditation or inheritance is particularly useful when certifying and accrediting SOA Infrastructure Services.

Describing the accredited system, application, or package of applications in terms of commitments, provisions, and obligations clarifies the roles of type accreditation and inheritance. The accredited system:

- Commits to providing specific IA controls to other systems, thus allowing those controls to be inherited.
- Is subject to provisions, that is, a specific set of conditions which hold for the technical, operational, and threat environment in which it is deployed. Some of the conditions could be that the accredited system, application, or package of applications inherits specific IA controls from another system.
- Imposes obligations on systems or applications that use it and thereby inherit its IA controls.

Note that using checklists is a sensible way to document provisions and obligations.

C&A of SOA infrastructure services—The SOA Infrastructure Services component provides infrastructure services that implement key, frequently used pieces of enterprise functionality. Security controls that may fall into this category include:

- Identification & authentication
- Authorization and access control
- Data confidentiality
- Data integrity
- Monitoring and audit

C&A options available in the DIACAP illustrate how enterprises might certify and accredit services falling in the SOA Infrastructures Services category.

In the DIACAP, the scope of C&A can be an Automated Information System (AIS) application, an enclave, an outsourced IT-based process, or an IT platform interconnection. C&A of a set of SOA Infrastructure Services could be handled in any of several ways:

- **Be treated as an AIS application**—This approach is relevant to SOA Infrastructure

Services that provide IA controls such as those shown in Figure 6. Treatment as an AIS application allows for type accreditation, wherein the AIS is approved for use in any environment that meets the conditions stated in the accreditation statement. Any system or application that uses the services as specified in the accreditation statement and that is deployed in an environment that meets the stated conditions inherits the security controls that the SOA Infrastructure Services provide.

- **Be part of an enclave**—When security controls extend to the limits of a clearly defined logical or physical boundary, the IA controls that the SOA Infrastructure Services provide could be certified and accredited in the context of an enclave C&A. Any system or application in the enclave that used the services would inherit the security controls that the SOA Infrastructure Services provide. However, deployment of those services in a different enclave would entail additional C&A.
- **Be an outsourced IT-based process**—For example, credentialing services, as defined in the Liberty Identity Assurance Framework,[31] could be outsourced. This approach could be relevant to some unclassified Government systems.

As noted earlier, type accreditation of a subsystem consisting of a set of SOA Infrastructure Services as an AIS application is consistent with the SOA design philosophy. The subsystem commits to providing specified IA controls via services as certified. The subsystem will meet its commitments subject to provisions or stated assumptions, such as "The Hardware and Software Foundations shall provide a user directory." Any service that relies on the subsystem may be subject to a set of requirements or obligations as well, such as "The access control policy for an application service must be clearly stated, so that policy decision rules can be written." The subsystem owner then produces documentation that can be integrated into a C&A package for systems, enclaves, or platforms that are to make use of the subsystem.

Along with the documentation, all commitments, provisions, and obligations associated with the subsystem should be identified. Here "commitments" should be expressed as Security Control requirements written in standard form (i.e., compliant with relevant regulations, such as the DoDI 8500.02 in the case of DIACAP or NIST SP 800-53) to the extent possible, but adapted with SOA specific wording where necessary. Provisions and obligations should be stated clearly in order to facilitate the assessment of the configuration of any system into which the subsystem is integrated. This can be done by creating checklists. Ideally C&A would also be supported by tools for assessing the SOA configuration and policy templates for user-facing and data-facing services. An open issue with this approach is whether registries could be extended to include checklists, templates, or other ways of capturing provisions and obligations.

C&A for hardware and software foundations and SOA infrastructure—C&A of the Hardware & Software Foundations and SOA Infrastructure components could be type accreditations or could be part of the overall C&A effort associated with a system, platform, or enclave to be deployed. The former option is the easier of the two but is more rigid in that the system being accredited must meet the type's technical, operational, and threat assumptions.

C&A of these two subsystems must show that each component meets:

- The provisions as defined in the SOA Infrastructure Services accreditation package.
- Its commitment to satisfactorily implement the security controls allocated to it as part of overall system C&A.
- The obligations imposed by the other subsystems.

Recall that although a system may have a service-oriented architecture, this does not guarantee that its security controls are in the SOA Infrastructure component. It may rely on the Hardware & Software Foundations component for some or even all of its security controls.

Examples of security controls that may be allocated to the Hardware & Software Foundations component include traditional firewall functionality or compliance with relevant configuration guidelines. Consequently certification activities generally consist of configuration assessments. The certification process should include clear statements of constraints to be placed on other subsystems that will use the Hardware & Software Foundations component as well as evidence that the other subsystems do, in fact, meet these obligations.

Components of the SOA Infrastructure subsystem are increasingly taking on the responsibility for implementing certain security controls. Enterprise Service Buses frequently incorporate security functionality, and the capabilities provided within XML gateways are expanding. In cases where security controls have been allocated to the SOA Infrastructure, certification must include evidence that the designated controls are fulfilled. And, of course, the certification process should include clear statements of constraints to be placed on other subsystems that will use the SOA Infrastructure component as well as evidence that the other subsystems do, in fact, meet these obligations.

Ideally, accreditation of both the Hardware & Software Foundations and SOA Infrastructure components is performed once, as part of the C&A effort of the overall system/platform/enclave. However, one of the stated advantages of using SOA components is that they can evolve internally without service use disruption. If such changes are materially security relevant and the existing risk not acceptable, a new C&A should be performed.

C&A for application services—Application services are the core building blocks of business functionality and mission utility. An application service that will be deployed in multiple locations or enclaves, as would be expected in an SOA-based system, should be type accredited when possible, as with the DIACAP process.

If the application service does not implement security controls, its C&A can be minimal. Certification of an application service provides evidence that all obligations imposed by the underlying SOA subsystems on which it is deployed, including the SOA Infrastructure, SOA Infrastructure Services, and Hardware & Software Foundations, are met. Thus certification ensures that security functionality is properly used by the application service.

In cases where some security functionality is provided by a service for its own use, certification must evaluate the effectiveness of these capabilities. Examples of such functionality include attribute-based data filtering, application-internal audit data filtering, and application-internal audit.

Tying it all together—The approach to C&A recommended here aligns the business or mission goals of service-oriented architectures with an enterprise's risk management goals and addresses the challenges identified earlier. Exploiting inheritance and type accreditation of subsystems allows different subsystems to be deployed or updated at different times. As a new subsystem is introduced, checklists—of provisions and obligations that apply to subsystems on which the new subsystem depends and on subsystems or applications that depend on it—can be used to determine whether any additional C&A efforts are needed.[32] This tactic allows for the incremental certification and deployment of application services over time, a key demand being placed on SOA designs. In addition, type accreditation facilitates replication of services in multiple enclaves. However, even using these tactics, SOA designs incorporating unfettered dynamic discovery of services will likely be precluded. At this time there appears to be no simple way to dynamically discover a service while simultaneously meeting C&A requirements.

Other issues—Even if the tactical approaches to SOA C&A described earlier are successful, organization management and security architects still need to be aware of other concerns. In particular, current C&A packages generally include specific identification of the system's user community and system boundaries, yet these are often abstractions or lacking in precise definition in service-oriented architectures. Furthermore C&A involves meeting requirements statements about security controls that do not specifically address SOA-based systems.

The issue of end-user identification can be addressed in part through documented constraints on the SOA Infrastructure Services provided. For instance, the I&A service would only grant access to end users of type *t*, as defined to be acceptable to the Accreditor.

The issue of system boundaries is assisted in part by regulations. The NIST SP 800-37, for instance, allows for flexibility in defining the accreditation boundary. Forthcoming Committee on National Security Systems (CNSS) Instructions[33] will build on the 800-37 approach, and NIACAP and DIACAP retain the notion of type accreditation, which facilitates the C&A process for SOA-based system.

Requirements standards are also ill prepared for the C&A of SOA-based systems. Current security controls are written in an SOA-agnostic fashion and in some cases may implicitly assume an architecture that is not service oriented. Controls are typically defined in reference to a multi-user system managed as an integrated whole by a single organization and

program manager. So, tailoring of these requirements for an SOA-based system is often necessary. This may include a discussion of whether the control is meaningful and what the control definition means in an SOA context. Additionally defining SOA-specific security controls such as identity federation may be useful in moving through the C&A process. It is also important to interact early and frequently with the relevant certification authorities (CAs) and designated accreditation authorities (DAAs).

Key Observations

1. SOA-based systems incorporate many non-traditional approaches to IT delivery, including incremental deployments, support for unanticipated users, ill-defined boundaries, and incorporation of services out of the direct control of the system owner.
2. The C&A processes and procedures used for traditional systems often do not match well with how SOA-based systems are designed and implemented.
3. Three tactics are available to cope with the challenges of SOA C&A: scoping, type accreditation, and inheritance.
4. Scoping refers to the packaging systems, services, and components in such a way as to make SOA C&A feasible.
5. Type accreditation enables a package of components to be certified under a defined set of assumptions.
6. Inheritance allows a certified system to provide security controls to another.
7. Systems to be accredited should be described in terms of their commitments, provisions, and obligations. A system operates under commitments to provide specific security controls, subject to stated provisions, and requires that relying systems adhere to obligations.
8. The commitments, provision, and obligations should be documented. System owners can benefit from using checklists to help this documentation.
9. Organizations seeking C&A for an SOA-based system should expect to modify and fit standard security control requirements to meet the realities of the SOA design.
10. Early and frequent interaction with CAs and DAAs is prudent.

Summary

The security objectives for SOA-based systems are the same as those for their non-SOA system counterparts: confidentiality, integrity, access control, accountability, and availability. This paper discussed several challenges that arise in securing SOA-based systems, provided various design options, noted where design trade-offs exist, and made specific recommendations where warranted.

An important consideration for systems owners and security architects is that some security functions can be implemented as services. Security policy decision points, security policy or attribute administration, attribute or policy retrieval, and certificate validation all lend themselves to implementation as services within an SOA-based system.

The paper looked at data protection options: at the network transport layer or message layer (or some combination of these). Messages are ubiquitous in SOA designs, and how they are protected can have serious ramifications. Though message-level protection affords great flexibility, it usually involves some performance degradation.

Another architectural concern is where services are located. Typically there is a trade-off between the lessened administrative burden and cost of deploying enterprise services versus the possibility that the loss of a single enterprise service deprives consumers.

Implementing authentication and authorization functions as services allows these critical services to be standardized across an enterprise. The paper discussed the related matters of service chaining, PKI use, mutual authentication, and delegation, all of which should be addressed by security architects.

One potential benefit with SOA implementations is how well they fit with the goal of sharing information across organizational boundaries. As the paper points out, organizations should agree on how security controls will be deployed and which organizations have governance responsibilities.

System owners need to review their audit requirements vis-à-vis the usage pattern of SOA services. As usage patterns trend toward unpredictability, creating audit trails becomes increasingly difficulty.

Obtaining certification and accreditation for SOA-based systems requires that system owners accept

the fact that current C&A procedures were not developed with SOA designs in mind. The paper offers some advice on how to deal with this mismatch: careful scoping, type accreditation, and inheritance. Also, when security functions are implemented as enterprise services, the apparent burden of C&A might be substantially reduced. Enterprise services accreditation followed by compliance validation at consumer systems might yield substantial long-term benefits.

Glossary [34]

Access Control—Protection of system resources against unauthorized access; a process by which use of system resources is regulated according to a security policy and is permitted by only authorized entities

Authentication—The process of verifying an identity claimed by or for a system entity.

Authorization—An "authorization" is a right or a permission that is granted to a system entity to access a system resource.

Availability—The property of a system or a system resource being accessible and usable on demand by an authorized system entity, according to performance specifications for the system; i.e., a system is available if it provides services according to the system design whenever users request them.

Confidentiality—The property that information is not made available or disclosed to unauthorized individuals, entities, or processes.

Identification—An act or process that presents an identifier to a system so that the system can recognize a system entity and distinguish it from other entities.

Integrity—The property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner.

Security Audit Trail—Chronological record of system activities that is sufficient to enable the reconstruction and examination of the sequence of environments and activities surrounding or leading to an operation, procedure, or event in a security-relevant transaction from inception to final results.

References

[1] "Leveraging Federal IT Investment—Using Service-Oriented Architecture" (SOA: An Analysis of SOA's Value Proposition for Federal Senior Leadership Teams, November 2008, MITRE Technical Report MTR080331).

[2] Cser, Andras, Forrester Research, April 2, 2008, Identity-As-A-Service The Evolution Of Identity Management http://www.forrester.com/Research/Document/0,7211,43824,00.html.

[3] In this paper, "systems" are the active parts of a computing environment and include software, applications, services, and processes along with the hardware on which these reside.

[4] The weather service, flight plan service, and legacy application are placed as shown as examples. Actual realizations could be characterized differently, depending on implementation details.

[5] Dierks, T. and C. Allen, January 1999, RFC 2246, The TLS Protocol Version 1.0.

[6] WS-Security is a standard for enabling security for communications in web services environments. http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[7] TLS and its precursor, SSL, are the standard protection mechanisms between browsers and www sites.

[8] Defined by a suite of Request for Comments (RFCs) http://www.ietf.org.

[9] XML Signature Syntax and Processing (2nd Ed.), June 2008 http://www.w3.org/TR/xmldsig-core/

[10] Betancourt, John, IBM, July 2007, Create a roadmap for securing your large-scale SOA application; Ten steps toward better SOA security http://www.ibm.com/developerworks/library/ar-soasec1/].

[11] Kevin Smith, May 2008, Avoiding SOA Security Anti-Patterns: Practical Planning for Success http://www.soainstitute.org/articles/article/article/avoiding-soa-security-anti-patterns-practical-planning-for-success.html

[12] Throughout this paper, "local" and "enterprise" are relative terms. "Enterprise" refers to the greatest network or organizational span relative to service provision. Enterprise services can reach anywhere within that span, whereas local services can reach only a subset of the enterprise span.

[13] Federal Information Processing Standards Publication 197, November 26, 2001, ADVANCED ENCRYPTION STANDARD (AES).

[14] In this paper, "enclave" refers to a physical boundary containing one or more systems which provide IT services primarily within the boundary. A corporate campus and a military base are examples.

[15] An exception being Internet-facing services designed for unrestricted use.

[16] http://www.oasis-open.org/specs/

[17] Shirey, R., May 2000. RFC 2828, Internet Security Glossary.

[18] eXtensible Access Control Markup Language Version 2.0, Oasis Standard, February 1, 2005.

[19] Baer, Tony, August 2007, "SOA Security: Centralize and Integrate," SOA Security at SAIC.

[20] Framing agreements in terms of commitments, provisions, and obligations arises again under Certification and Accreditation for an SOA-Based System.

[21] Weill, P., and J. Ross, 2004, IT Governance: How Top Performers Manage IT Decision Rights for Superior Results, Harvard Business Publishing.

[22] Web Services Policy 1.2 - Framework (WS-Policy), April 2006. http://www.w3.org/Submission/WS-Policy/

[23] Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl

[24] UDDI 3.0.3, October 2004.
http://www.uddi.org/pubs/uddi_v3.htm

[25] For example, NIST Special Publication 800-53, Revision 2, December 2007, Recommended Security Controls for Information Systems

[26] David Linthicum, March 2005, Why Orchestration Defines Your SOA.
http://blogs.ittoolbox.com/eai/cto/archives/why-orchestration-defines-your-soa-3685

[27] http://www.cio.com/article/104007/How_to_Navigate_a_Sea_of_SOA_Standards

[28] The Committee on National Security Systems (CNSS) is in the process of redefining overarching process and control requirements. Existing catalogs of IA controls—DoDI 8500.2, DCID 6/3, and NIST 800-53—will be mapped to the controls defined by CNSS. Similarly, existing C&A processes will be aligned with the overarching process.

[29] IBM Working Paper,
https://acc.dau.mil/GetAttachment.aspx?id=140000&pname=file&lang=en-US&aid=27194

[30] DCID 6/3 - Protecting Sensitive Compartmented Information within Information Systems Manual

[31] Liberty Identity Assurance Framework, Version 1.1
http://www.projectliberty.org/liberty/content/search?SearchText=credential+framework

[32] Service commitments on which C&A is based should be specified in SLAs or MOUs.

[33] http://www.cnss.gov

[33] Definitions from R. Shirey, Internet Security Glossary Version 2, RFC 4949 August 2007.

Contributors

Additionally, the editor would like to thank the following people for their contributions and feedback.

Deb Bodeau

Don Faatz

Marie Francesca

Ed Kenney

Andrew MacBrien

Mindy Rudell

Larry Pizette

Aaron Temin

Jackson Wynn

www.mitre.org

Approved for Public Release
Distribution Unlimited
Case Number: 10-0002
Document Number: MTR090000

MITRE