Designing Dependable Service Oriented Web Services Security Architectures Solutions

Dr.D.Sravan Kumar[1] and M.Upendra Kumar[2]

[1]Principal and Professor CSE KITE Womens College of Professional Engineering Sciences Hyderabad
[2]Research Scholar JNTU Hyderabad and Associate Professor CSE MGIT Hyderabad

***Abstract*—System Security Architecture from a software engineering viewpoint imposes that strong security must be a guiding principle of the entire software development process. It describes a way to weave security into systems architecture, and it identifies common patterns of implementation found in most security products. The security and software engineering communities must find ways to develop software correctly in a timely and cost-effective fashion. There's no substitute for working software security as deeply into the development process as possible. System designers and developers must take a more proactive role in building secure software. The root of most security problems is software that fails in unexpected ways when under attack. The enforcement of security at the design phase can reduce the cost and effort associated with the introduction of security during implementation. At the architecture level a system must be coherent and present unified security architecture that takes into account security principles (such as the least privilege). In this paper we want to discuss about different facets of security as applicable to Service Oriented Architectures (SOA) Security Architecture implementations. First we examine the security requirements and its solution mechanisms. In the context of Web Services, the predominant SOA implementation standard has a crucial role to play. The Web Services architecture is expected to play a prominent role in developing next generation distributed systems. Building dependable systems based on web services architecture is a major research issue being discussed. Finally, we provide a case study of Web Services Security Architecture, enhancing its security pertaining to Web 2.0 AJAX (Asynchronous JavaScript and XML) and its Security encryption of data using MD5algorithm.**

***Index Terms*—Security Architectures, Designing Dependable Architectures, Service Oriented Web Services Security Architectures, Web 2.0 Services AJAX Security**

I. Introduction To Security Architectures

Architecture, whether system or application, are composed of abstractions (interfaces) and their implementations. Security Architectures are architectures which enable implementations that are resilient to an appropriate and broad-based spectrum of threats [1]. An evaluation of Security Architectures requires understanding these threats; the tradeoffs between different system goals, including between security and non-security goals; the long-term appropriateness of its interfaces; and the implementations it allows. The best interfaces are those that capture the most important issues, enable different implementations, and are flexible enough to adapt (are be adapted) to different threats. Two well-known issues are particularly important: First, complexity is a source of security holes. Second, security is a matter of the weakest link. Because of the need to balance of complexity versus protections, these tradeoffs are often controversial. Other tradeoffs include performance, usability, and flexibility. The design and evaluation of Security Architectures is of fundamental importance to security and yet many of our fundamental architectures were created when security was less appreciated and less well understood. Since it is notoriously difficult to add security after the fact, our systems are far too susceptible to attack. Moreover, architectures, because they are broad based, are difficult to understand [2]. Table I below shows the security life cycle phases with their definitions.

TABLE I. SECURITY LIFECYCLE PAHSES

Life Cycle phase	Definition	
Design	From initial idea to design specs	Subtly alter system specification to create a flaw
production	From build-to specs to roll-out	Substitute security-critical chip on production line
Deployment	From roll-out to transit to delivery to user	Substitute system unit while in transit with bogus unit
Operation and maintenance	From delivery to maintenance to retirement	Insert malicious code into application,OS,or network
Destruction	From retirement to destruction	Extract stored key from unit to read back-traffic

II. Service Oriented Security Architectures

Service-Oriented Architecture (SOA) is "a paradigm for organizing and utilizing Distributed capabilities that may be under the control of different ownership Domains."[3]. i.e. SOA is collection of services, where these services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. A service is a component participating in a service-oriented architecture that provides functionalities or participate in realizing one or more capabilities. For the last few years, a rise has been observed in research activity in SOA, with applications in different sectors . Several new technologies have been introduced and even more are being currently researched and aimed to the future. Service oriented mentality with the purpose of lessening the issues of clients and companies, students and teachers, citizens and Government companies alike has the most influential approach from software engineering point of view that belong either to

the academic or to the industrial field. Refer Figure 1 for Web Service Security Architecture.

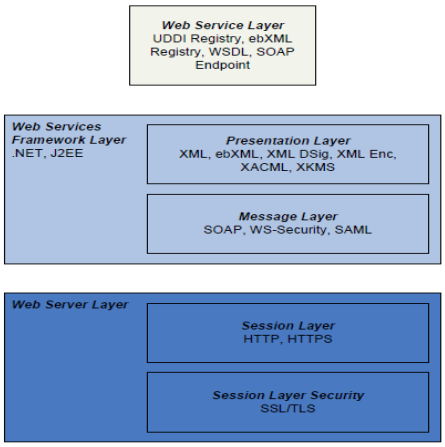

Figure 1.Web Service Security Architecture

Software Architecture has been emerging as a discipline over the last decade. A System Software Architecture describes its coarse-grained structure and its properties at a high level. As long as the technology supports those structures and properties, the technology can be considered to implement the architecture, for instance Jini is a technology that supports Service-Oriented Architectures because it supports the properties of Service-Oriented Architecture. It is important to apply the concepts of Software Architecture to any new technology to take full advantage of it. SOA is implemented by technologies other than Web Services, but the terms and concepts have gained recently because of Web Services. For instance, the computer industry has used the term service for about two decades to describe various platforms. Some of the characteristics of SOA are supported better by certain technologies than by others. For instance, CORBA (Common Object Request Broker Architecture) and Jini are less interoperable than Web Services, but Jini excels in other properties (though this is arguable), such as discovery. Interface design is perhaps the most difficult part of designing services in SOA. The modularization techniques practiced for decades still apply to services. Service design is even more difficult, because the domain a service supports is not limited to a single application. Therefore, it is best to perform modularization starting with a conceptual model of the business rather than of a single application. If the interface design is done well the services are more likely to be reusable in other application, and organizations will realize a higher return on their investment. Web Services are refocusing organizations on the concepts of SOA. Although highly reusable, loosely coupled architectures have been a goal for many organizations. Web Services are fostering interest in and provides the technology to implement SOA that enable them to realize their vision.

Data Services will be integral to designing, building and maintaining SOA applications. A Data service enables business processes to access and manipulate SOA applications Business Objects. A typical data service will provide a set of operation that encapsulate different ways to access business objects of a given type, to simplify data access for consumers of the services. Further, it can relate to other data services in accordance with relationships among the various business objects. Data services thus enrich the SOA model by letting application developers more easily and rapidly understand the enterprises sea of services, facilitating service discovery and reuse.

With web services and service oriented architecture software development is in the third phase of evolution that began with object oriented programming and progressed to component based programming. In object oriented and component based programming, security designers could relay on common languages, security models, and technologies in a distributed system to secure both the client and servers transaction and end points for example, EJB clients and servers can assume and use a common J2EE security standard for authentication and authorization for both its client and server. However in web services and service oriented architecture, systems are loosely coupled clients and servers may be written in different languages and running on disparate operating system. Service oriented architecture agrees upon interface and data schema, but the implementations may vary from client to server, and services may be chained creating peer to peer model. As programming models and implementations change, the security assumptions and designs must be refreshed and updated to manage the emerging threats that results from new architectures

The primary security mechanisms deployed today relay on notions of perimeter and centralized security models. However the nature of business is moving rapidly towards decentralized “intertwingled-ness” (non-hierarchical connectivity) and perimeters are eroding. Centralized security systems don’t apply in SOA’s decentralized peer to peer architecture. Security design assumptions based on outmoded technology create brittle and ineffective systems when deployed in a SOA paradigm. Malicious attackers exploit the seams left between the existing security mechanisms deployed based on outmoded assumptions and reality of the threats to the connected systems on the ground. Assuring multiple qualities attributes – such as reliability, availability, performances, security and real-time response – for a variety of critical applications makes it essential to develop practical techniques for implementing high-assurance service-oriented systems. Schemes for intrusion detection yield clues to making data safer in the future. It helps determine what systems were attacked and exactly how the attacks were made. Data collected aids in tracking the source of an attack, which may prove helpful in identifying the attacker.

Architecture issues: The views describe a way of seeing security architecture across a complex system to make and convey security design decisions. The software security space contains issues that are still being worked to achieve optimal effectiveness [4].

XML Security: Research has shown various flaws with XML security related to its reliance on XML for encryption and signature as well as replicating a number of problems in the legacy technologies. Since a large number of emerging security solutions, particularly WS-* rely on XML security mechanisms it is worth revisiting this dependency to see if XMPP or other technology can remedy these issues.

The SOA provides an abstraction utility that is characterized to be autonomous, well-defined and self-contained. Research defines the building blocks of a security reference model composed out of Processes Domain View, Security Assurance View and Survivability Management view. We build a security attributes organizational model based on security process states and security attributes requirements. The proposed architecture is based on SOA reference model with mappings of SOA dimensions into security requirements attributes.

III. WEB SERVICES SECURITY ARCHITECTURES

A Web Service is a software component or system designed to support interoperable machine or application-oriented interaction over a network. [5] A Web Service security has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP (Simple Object Access Protocol) messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. Web Services Security (WS-Security) is a mechanism for incorporating security information into SOAP messages. WS-Security uses binary tokens for authentication, digital signatures for integrity, and content-level encryption for confidentiality.

A Web Service is a software entity deployed on the Web whose public interface is described XML (eXtensible Markup Language) . It can interact with other systems by exchanging XML-based messages, using standard Internet standard protocols. The Web Services definition and location (given by a Uniform Resource Identifiers URI) can be discovered by querying common Web Service Registries. Web Services can be implemented using any programming language and executed on heterogeneous platforms, as long as they provide the above features. This allows Web Services owned by distinct entities to interoperate through message exchange.

As Web services become more widely adopted, developers must cope with the complexity of evolving trust negotiation policies spanning numerous autonomous services. The Trust-Serv framework uses a state-machine-based modeling approach that supports life-cycle policy management and automated enforcement.

The Web Services architecture is expected to play a prominent role in developing next generation distributed systems. It targets the development of applications based on XML-related standards, hence easing the development of distributed systems through the dynamic integration of applications distributed over the Internet, independently of their underlying platforms. Web Services Security Architectures have three layers viz. Web Service Layer, Web Services Framework Layer (.NET or J2EE), Web Server Layer. Web 2.0 increases web based access to data processing particularly on the client side that enables web applications which contain enriched functionality. Web 2.0 technologies have wide range of technologies and protocols which enable Web architectures greater access to data and functions. The technologies include AJAX (Asynchronous JavaScript and XML), XML, JSON(JavaScript Object Notation), SOAP (Simple Object Access Protocol) and WSDL(Web Services Description Language), REST Web API's, Microsoft Silver light, RSS, RDF, and Atom. Web 2.0 vulnerabilities include XML, JavaScript, RSS, AJAX, SOAP, JSON, WSDL, in decreasing order of their attack statistics.

In this research, we want to implement security tools to Web Services Architecture in terms of layers and above attacks [6]. Initially, building web services by combing protocols like REST and WS-* will be studied. Later This Web Services will be secured by adding policy, custom authentication, creating client Security Tools, .NET Cryptography, Securing Data Access, and Protecting Code. Etc.

Services must be designed and composed in a secure manner. In particular, we are concerned with safety properties of service behavior. Services can enforce security policies locally and can invoke other services that respect given security contracts. This call-by-contract mechanism offers a significant set of opportunities, each driving secure ways to compose services. We can correctly plan service compositions in several relevant classes of services and security properties. We can propose a graphical modeling framework based on foundational calculus. This formalism features dynamic and static semantics, thus allowing for formal reasoning about systems. Static analysis and model checking techniques provide the designer with useful information to assess and fix possible vulnerabilities.

Securing Web Services Architecture: An element of Security for Web Services consists of Authentication, Authorization, Integrity, Non-repudiation, Confidentiality, and Privacy. Properties of Secure Software for Web Services are Predictability of operation, Simplicity of software design and code, correctness, and safety. The challenge for secure web services has these dimensions: Secure Messaging, Protection of resources, Negotiation of contracts, Trust management. Common attacks against Web Services include: Reconnaissance attacks, Dictionary attack, Forceful browsing attack, Directory traversal attacks, WSDL Scanning, Sniffing, Privilege escalation attempts, Format String attacks, Exploiting unprotected administrator interfaces, Attacks on confidentiality, Registry disclosure attacks, attacks on integrity: Parameter tampering, coercive parsing, schema poisoning, spoofing of UDDI/ebXML messages, Principal spoofing, Routing detours, External entity attack, cannonicalization, intelligent tampering and impersonation, Denial of Service attacks, Flooding attacks, Recursive payloads sent to XML parsers, Oversized payloads sent to XML parsers, Buffer overflow exploits, Race conditions, Symlink attacks, Memory leak exploitation., Command injection, Structured Query Language injection, XML injection, Malicious code attacks, URL String attack, Parameter Tampering, Cross-site scripting, Session Hijacking, Malformed content, Logic Bombs Trapdoors Backdoors[7].

Several standards are establishing a framework for integrating security into domain-specific XML-based applications. WS-Federation standardizes the way companies share user and machine identities. Risk management is not well understood within the Information Security community. The Security and Software Engineering communities must find ways to develop software correctly in a timely and cost-effective fashion. Overly broad and vague laws have created a cloud of legal uncertainty over an important area of security research and engineering.

Security requires an end-to-end perspective and not just a point-to-point one. It is not simply the exchange of data between the client and the server that is important, but instead the entire path that the data takes. This includes not only technologies, but also operational processes. Do not encrypt the entire message. Due to the overhead of encryption and decryption, only encrypt what needs to be encrypted. Encrypt data meant for different people suing different keys. The advantage of using XML Encryption is that it supports both of these requirements. Inline signatures with the information that they sign. Signed documents are important not only during transmission between parties, but also as a means to prove and enforce accountability and liability. To do so, signed documents must be easily archived, so that both the contents of a document as well as its signatures can be easily retrieved at a later time. XML Digital Signatures supports inline signatures and also allows different signatures for different parts of a document. WS-Security is emerging as the defacto standard for a comprehensive framework for Web Services security.

Table II below summarizes the different security approaches for securing Service Oriented Web Services Architectures.

TABLE II. DIFFERENT SECURITY APPROACHES

Approach	Example	Advantages	Disadvantages
Network	• Router • Firewall • Packet Filter	• Limits access to machines that are authorized to operate within a particular network boundary. • Blocks traffic based on IP addresses, protocols, and port assignments. • Is transparent to applications.	• Cannot inspect application traffic. • Does not limit eavesdropping • Provides authentication and authorization of host machines only.
Transport	• SSL • TLS	• Limits access to resources that are authorized to use a service. • Blocks traffic based on public key certificates. • Digitally encrypts the transmission of data • Offers point to point security	• Does not provide end-to-end security. • Does not make security credentials available to application. • Provides all-or-nothing access control only
Application	• Custom application • Software Module	• Limits access to resources that are authorized to use a service • Blocks traffic based on message contents. • Digitally signs messages. • Digitally encrypts messages.	• Requires knowledge of the application protocols. • Must be individually built or customized for each type of application
		• Offers end-to-end security • Makes credentials available to applications.	

IV. Designing Dependable Solutions

Key analysis and design consideration for security of architectures deals with, “How well the system authenticates the users and protects the application and data elements?” [8]. Various Software Engineering paradigms exist such as model-driven, aspect-oriented and agent-oriented. New methods and techniques are required should support the formal (and simultaneous) modeling, reasoning and analysis of security and functional requirements, and transformation of such (security and functional) requirements to a design that will satisfies them along with support for traceability and validation of the proposed solution.

Important design principle is principle of Least Privilege: It states that a subject should be given only those privileges that it needs in order to complete its task. There are tasks that need to be performed only at an elevated privilege. Any computer program must always remain in a least-privileged state. When there is a need, it will elevate the privilege only for the duration needed. Once the privileged function is complete, it must return to the state of least privilege.

Tool support: A tool should not only support developers in modeling and reasoning about security (and functional requirements) during the analysis stage, but it should help to transform the requirements to design, check the consistency of the proposed solution and to validate the security functionalities of the proposed solution and to validate the security functionalities of the proposed solution and also validate the security functionalities of the proposed solution against the security solutions of the system.

Architecting Web Services Security: Security requires an end-to-end perspective and not just a point-to-point one. It is not simply the exchange of data between the client and the server that is important, but instead the entire path that the data takes. This includes not only technologies, but also operational processes. Do not encrypt the entire message. Due to the overhead of encryption and decryption, only encrypt what needs to be encrypted. Encrypt data meant for different people using different keys. The advantage of using XML Encryption is that it supports both of these requirements. Inline signatures with the information that they sign. Signed documents are important not only during transmission between parties, but also as a means to prove and enforce accountability and liability. To do so, signed documents must be easily archived, so that both the contents of a document as well as its signatures can be easily retrieved at a later time. XML Digital Signatures supports inline signatures and also allows different signatures for different parts of a document. WS-Security is emerging as the de facto standard for a apprehensive framework for Web Services security.

V. WEB 2.0 SERVICES AJAX SECURITY ARCHITECTURE CASE STUDY

Web 2.0 increases web based access to data processing particularly on the client side (AJAX Asynchronous JavaScript and XML) that enables web applications which contain enriched functionality [9]. Web 2.0 technologies have wide range of technologies and protocols which enable Web architectures greater access to data and functions. The technologies include AJAX (Asynchronous JavaScript and XML), XML, JSON (JavaScript Object Notation), SOAP (Simple Object Access Protocol) and WSDL (Web Services Description Language), REST Web API's, Microsoft Silverlight, RSS, RDF and Atom. Web 2.0 vulnerabilities include XML, JavaScript, RSS, AJAX, SOAP, JSON, WSDL, in decreasing order of their attack statistics.

We had implemented securing AJAX Web Services issues. In order to design a secure web tier for AJAX applications, we first need to study the architecture of the AJAX architecture. The client running in the user's browser makes requests to the server using Hypertext Transfer Protocol (HTTP). These requests are processed by the Web server processes, such as Servlets, dynamic pages, etc. The response time is returned to the client in the form of the streams of data. The web services or pages are accessed by the external entities, without any additional work on our part. It may be that we encourage outsiders to use our web services in this way, and we may even publish an API, as eBay, Amazon and Google among others have done. Even in this case, though, we need to keep security in mind. There are two things that we can do, the first one is to design our web services interfaces, or API in such a way that external entities would not be able to subvert the purpose of our web application, e.g. by ordering the products without paying for them(designing a secure web tier). The other one is to look at the techniques to restrict access to the Web services to particular parties. When we design a web application, we typically have an end-to-end workflow in mind. For example in a shopping site, the user will browse the store, add the items to their basket, and then proceed to checkout. The check out process on the other hand has its own well-defined work flow with the choice of delivery address, shipping options, and confirmation of order. As long as the application is calling the shots, we can trust that the workflow is being used correctly. If an external entity starts to call our web services directly, however, we may have problem.

We had implemented this example using third party services, creating a search control, configure it to search across Local search, Web search, Video, and Book search, and then place the cursor on our page. (Refer Figure 2 below). We are securing our browser using JavaScript, both for the same browser and cross browser. We are using the features of Google search API directly from the site of Google. The function onload is used to load the Google search as soon as the page gets loaded. First, we created a Google search control and assigned it a variable searchcontrol. Then we added the searchers required for our example. We have used only four searches and left the image, news and blog searches. If we want to include all the searchers, we can add searchers the same way. Then we set the Local Search center point, we have a method searchcontrol.draw( ). This method is used to display different draw modes – tabbed or linear. We have used the linear mode in this example. Next, we have the nsearchcontrol.execute( ) method, which is used to initialize the search, and some code for the presentation logic of the page that we can easily understand. We have initialized the search for Ajax.

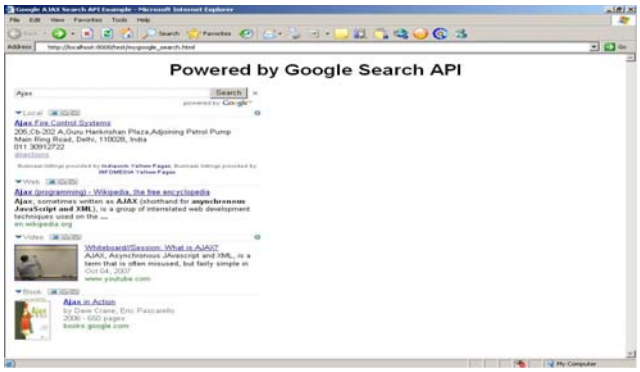

Figure 2.Initial Page of the mygoogle_search.html

This example generates a MD5 encrypted hash. (Refer figure 3 below) We show how to protect our data and to restrict unauthorized users to access it. The page contains two text boxes and a button for clearing the value entered by the user. The first text box is used to enter the password string (here it's Jntu University) and the other is used to show the encrypted form of the password using MD5 algorithm. An encryption algorithm will generate a random-looking, but predictable, output from an input string e.g. MD5 algorithm. It has a few key features that make it useful for security. First, MD5-ing a piece of data will always generate the same result, every time. Second, two different resources are monumentally unlikely to generate the same MD5 digest. Taken together, these two features make an MD5 digest (that is, the output of the algorithm) of a resource a rather good fingerprint of that resource. The third feature is that the algorithm is not easily reversible. The MD5 digest can therefore be freely passed about in the open, without the risk of a malicious entity being able to use it to decrypt the message. For example, the MD5 algorithm will generate the digest string for the password string entered by the user. We can encrypt the password on the client site and transmit the encrypted form to the server. The server on the other hand, will fetch the password from the database and encrypt it using the same algorithm. The server then compares the two encrypted strings. If the strings match successfully, it would allow the user to log on. However, we can't transmit the straight MD5 digest across the Internet in order to login. An unauthorized user may not be able to find the exact value for the password, but they would soon learn that the particular digest grants them to the user account.

Figure 3.Encryption of data using JavaScript (MD5 algorithm)

CONCLUSIONS

In this paper, we discussed research issues of integrating security into software architecture of Service Oriented Architectures Web Services Security Architectures, while providing dependable design solutions. Further work involves comparing SOA Web Services Security architectures of Sun ONE and Microsoft .NET. Dependability is the key factor if service-oriented computing is to become a success story even in critical areas such as public safety or air traffic control. In order to achieve the end-to-end vision of security, the individual security technologies embedded in the architecture layers need security management practices and a system design in which appropriate with necessary separations are part of the structural properties. Securing the architecture will therefore be a process (as is the case with a proper security strategy) that needs to encompass requirements from various types of architecture properties and different stakeholders. Future work for middleware is: the combination with service-oriented transaction techniques and the introduction of a security concept for the framework.

REFERENCES

[1] Gunnar Peterson, "Security Architecture Blueprint", Arctec Group LLC, 2007.

[2] Spyros T. Halkidis, Nikolaos Tsantalis, Alexander Chatizigeorgiou and George Stephanides, "Architectural Risk Analysis of Software Systems Based on Security Patterns," *IEEE Transactions on Dependable and Secure Computing*, vol. 5 no. 3, pp. 129–142, July-September 2008.

[3] Sasikanth Avancha, "A Framework for Trustworthy Service Oriented Computing", ICISS 2008, pp. 124 – 132.

[4] Ozgur Erol et al, "A Framework for Enterprise Resilience using Service Oriented Architecture approach", IEEE Sys Con 2009, 3 rd annual IEEE International Conference March 23 – 26 2009

[5] Anoop Singhal and Theodore Winograd, *Guide to Secure Web Services*. NIST Draft (800-95), September 2006.

[6] Massimo Bartoletti, Pierpaolo Degano, Gian Luigi Ferrari and Roberto Zunino, "Semantics Based Design for Secure Web Services," *IEEE Transactions on Software Engineering*, vol. 34 no. 1, pp. 33–49, January-February 2008.

[7] Ferda Tartanoglu et al,"Dependability in the Web Services Architecture", Architecting Dependable Systems, LNCS 2677, pp. 90 – 109, 2003

[8] Mouratidis and Giorgini, *Integrating Security and Software Engineering: Advances and Future Vision*. Idea Group Publishing Inc., 2007.

[9] Asoke K. Talukder and Manish Chaitanya, *Architecting Secure Software System*. CRC Press, 2009.