Security Based Service Oriented Architecture in Cloud Environment

Asha N. Chaudhary, Prof. Hitesh A. Bheda
M.tech (CE) Researcher Scholar RK University, India.
Department of computer Engineering. RK University, India.

ABSTRACT
Service Oriented Architecture is appropriate model for distributed application development in the recent explosion of Internet services and cloud computing. SOA introduces new security challenges which are not present in the single hop client server architectures due to the involvement of multiple service providers in a service request. The interaction of independent services in SOA could break service policies. User in SOA system has no control what happens in the chain of service invocations. Even if the establishment of trust across all involved partners is required as a precondition to ensure secure interactions, still a new end to end security auditing mechanism is needed to verify the actual service invocation. We provide solution for end-to-end security auditing in service oriented architecture. This security architecture introduces two new components called taint analysis and trust broker. It also taking advantage of WS-security and WS-Trust standards. These components maintain session auditing and dynamic trust among services. The solution of these services allows auditing of inheritance services without modification. We also implemented model of the future approach. We also established its efficiency in Amazon EC2 and multi tenancy cloud computing infrastructure.

Keywords: Service Oriented Architecture, Cloud Computing, Web Services, Security, Multi tenancy, Performance.

1. INTRODUCTION

Cloud computing means is a way of using computational resources such as storage, operating systems etc. Which are located remotely and are provided as a service over internet [1].The service oriented architecture is an idea of received significant attention and concern from the software design and development [2].

In software engineering, service oriented architecture is a new model in which is characterized by loose Coupling among software components, called services. SOA permit fast design of new applications by composing smaller special purpose and mixed services [3]. To ass mixed services components in both project and military environment, SOA can provides as the unifying layer [3]. Web service is a proven manufacturing technology that can be used to implement SOA application.

The basic necessities of SOA are: (1) the user must be able to control between different clouds as long as they are well-matched. An example would be if a client running an OS on an IaaS cloud. They should be capable in the direction of convey their transformation to the new cloud provider they want to control to [4]. (2) The user must be wanted to create a group of resources. An instance would be two cloud providers work jointly at providing their mutual resources through the same source [4].

Due to a sequence of principles that have been created based on standard extensible Markup language, web services allow interoperability of applications [2]. We can say that the most important advantage of using the model of SOA is interoperability which is achieved by the use of typical XML, which permit not only communication of straight usage in the web, but communication between devices ranging from small sensor to a complicated family machine, marketable or manufacturing [2].

To ensure security in this surroundings, new security mechanism must be measured, such as[5],[1]WS-security is a standard of Organization for Advancement of Structured Information Standards in order to SOAP messaging security and providing honesty and confidentiality.[2]Security Assertion Markup Language is another OASIS standard based on XML for exchanging security information.[3]Web Services Business Process Execution Language is an XML based language that is used to organize web service in single business process.

The SOAP-Simple Object Access Protocol is main important protocol which is used for web service to connect and is transported over the HTTP-Hyper Text Transfer Protocol [6]. Due to be short of end to end authentication and authorization, security is a demanding matter in service oriented
architecture. Attackers are not able to stop unnecessary interception of messages. It is also not possible to secure the unidentified third parties in SOA because of the architecture’s open nature [3]. Present Service Oriented Architecture security solution and web service security standards have the following restriction: (1) Web service principles are concentrating on transaction between only two communicating service end points. Web service standards do not consider service composition. (2) Outside services are not confirmed or validated dynamically means uniformed assortment of services by user. (3) User has no manage on outside service invocation through a service in another service domain. (4) Violation and malicious activities in a trusted service domain remain hidden.

The remainder of this paper is organized as follows: in section II we sketch out the proposed architecture for SOA system. In section III, we explain our model implementation. It also concludes a security and performance evaluation of the model. In section IV, we explain future work. In section V, we explain related work and finally in section VI, we explain conclusion of the paper.

II. PROPOSED ARCHITECTURE FOR SOA SYSTEM

2.1 End-to-End SOA Structure

The end-to-end SOA architecture consists in two steps: (1) client build request to the initial trusted domain (2) that services can make a service call to another service from trusted domain or an untrusted public domain.

Taint analysis and trust broker are two new components in current end-to-end security auditing architecture. Trust broker maintain information about trustworthiness of services and categorizes them. Trust broker is also used for dynamic validation and verification of services and keeps track of history of service invocation. Taint analysis module intercepts the communication.

Figure 1 End-to-End Service Oriented Architecture

In figure 1, the information sequence depicted is as follows:
1) UDDI (Universal Description, Discovery and integration) Registry request.
2) Forwarding the service list to Trust Broker and receive a categorized list.
3) Invoking a selected service.
4) Second invocation by service in domain A.
5) Invoking a service in public service domain.
6) End points reply to user.

2.2 Integration of Web Service Standards

The advantage of Web Service standard in our model which is used to achieve end-to-end security in our system: WS-Security and WS-Trust.

2.2.1 WS-Security

WS Security model specifies how integrity and confidentiality can be enforced on message, and allow the communication of various security token formats, such as SAML (Security Assertion Markup Language) and X.509 [9]. Its main focus is the use of Extensible Markup Language (XML) signature and Extensible Markup Language (XML) encryption to provide end-to-end security. WS-Security incorporate security features in the header of a SOAP (Simple Object Access Protocol message), specifying how to sign and encrypt SOAP (Simple Object Access Protocol) messages. There are two ways to handle the record management by WS-Security. First it defines a special element, Username Token, which pass the username and password if the web service is using custom authentication. WS-Security also provide binary authentication tokens such as Kerberos Tickets and X.509 certification: Binary Security Token. Figure 2 explain the flow of WS-Security.
The implementation of WS-Security uses the Apache CXF framework leveraging WSS4J to provide WS-Security functionalities. WSS4J security is triggered through interceptors that are added to the services and clients.

2.2.2 WS-Trust

It defines the concept of security token request and response message as well as ways to establish, assess the presence of, and broker trust relationships between participants in a secure message exchange.

Elements of WS-Trust:

1) Security Token Service is a web service that issues, cancels, renews and validates security tokens as defined in the WS-Security specification.
2) Security token request and response message format.
3) Key-exchange mechanism.

The implementation of WS-Trust within web-service libraries which provided by vendors or by open source cooperative efforts.

2.3 Trust Broker Formation

Trust Broker is trusted third party accountable for maintain end-to-end security in a chain of service invocation based on request of a client. It can also mediating security serious interactions between clients and services. The most important three function of Trust broker is as follows:

1) Trust Broker maintain list of licensed services.
2) Trust Broker evaluates the trust level of given service using formula which are integrate various parameter like history of communication with that service. And
3) Trust Broker maintains an end-to-end session of service invocation where the different services invoked from the start to the end of that session are logged by Trust Broker.

The Trust Broker was implemented as a web service in the java 7.0 platform and deployed on boston.cs.purdue.edu. Trust Broker which stores all data concerning session and services in MySQL database that is setup on same machine. The Trust Broker web service offers the following public method:

1) Get Trust Level (servicekey): This method gives the key of a service which is registered in UDDI and returns the trust level which is calculated by the trust evaluation module.
2) Create Session (trustclass, invokedservice): This method returns a single session identifier which needs to be along from client to the invoked service and from one service to other in the whole chain of service invocation.
3) Get Session History (sessionID): This method returns the register of warnings which is used for service invocation for session identified with sessionID.
4) Remove Session (sessionID): This method removes the session identifier with session ID from the trust Broker database.
5) Session Feedback: This method is used to connect taint analysis module with trust broker.
2.3.1 Trust Broker Database

The trust broker database consists of two tables: Services which are used to maintain the trust levels and certification information of services. Client Sessions which are used to maintain an end-to-end session of service invocations for a client.

- **Services**
  - Service_Key (String)
  - Trust_level (Double)
  - Certified (Boolean)

- **Client_Sessions**
  - Session_id (String)
  - SessionInvocation (String)
  - Trust_class (enum)
  - Session_history (String)

2.3.2 Trust Evaluation module

The trust evaluation module of the trust broker which is used to calculate trust level of given service that is based on three things:

1) History of earlier service runs.
2) Feedback from taint analysis module.
3) WS-supports specified in service level agreement.

TEM queries the UDDI for calculating trust value of service which is used following equation:

\[ Ts(t) = \beta \ast [\alpha \ast Ts(t-1) + (1 - \alpha) \ast F] + (1 - \beta) \ast L \]

Where \( \alpha < 0.5 \), \( \beta \) is the constant which is weight for the properties of the service. For the model system, the values for these constants were chosen arbitrarily so experiments should be performed to determine the optimal values for them. The trust value \( Ts \) for a service s, F is feedback parameter is service level agreement trus value.

In this equation feedback parameter has values in the interval \([-1, 0)\) when the services in question misbehaves and values in the interval \((0, 1]\) when the service behaves as promised. The result of this equation in the interval \([0,1]\).

2.4 Taint Analysis Module

The work of this module monitors the activity of services at runtime. It also examines the data replace between them to identify certain events.

2.4.1 Monitoring Runtime Services

One of the major design requirements of taint analysis module is transparency to the user means users are not required to change their programs or deployment. If we want to achieve this goal, program instrumentation and extra instrumentation are automatically added service implementation.

2.4.2 AOP Framework

Aspect Oriented Programming [7] model define some specific PCD (point cut designator) which are included in program execution. The fundamental point cut designators are chosen somewhat practically, they must be really helpful to an aspect programmer, but they must also be comparatively useful to implement in the AOP system.

2.4.3 Features of JBoss AOP

JBoss Aspect Oriented Programming [8] is used to operate an event framework.
JBoss Aspect Oriented Programming also used to provide dynamic AOP and hot deployment. JBoss Aspect Oriented Programming framework used to support both compile time and runtime class instrumentation.

2.4.4 Implementation of Taint Analysis

Implementation of Taint Analysis, we selected JBoss AOP framework. When we use JBoss AOP, we can almost all classes and methods in the JBoss AS/ESB servers. This mechanism is very efficient by using granular point cut. We implement communication methods inside an action pipeline.

As shown in the figure 4, all external service invocation are monitored and reported to the trust broker. Monitoring services are done for two activities. First is to check the compliance of those domains to their registered SLA agreement as advertised in the public UDDI registry. Second is the utilization of their data into the trusted service domain. The function of reporting to the TB is accomplished by web service invocation to the TB server. Trust Broker invokes the sessionFeedback() method.
Following method are used when we want to realize the connection between trust brokers and taint analysis. This method is also used the API of TB was extend.

Session Feedback (sessionID, invoker Service, invokedService):- This method are used for the taint analysis module for service reports to trust broker an invocation of invoked service by invoker Service for the session identified with session ID.

III. RESULTS AND DISCUSSION

This section provides the details of prototype evaluation and measurements.

3.1 Security Evaluation

SOAP messages are prone to attacks that can guide to several consequences such as unauthorized access, disclosure of information which is based on an on-the-fly modification of SOAP messages, referred as XML rewriting attacks [10]. The Denial of Service attack technology has continued to evolve and continues to be used to attack and impact internet infrastructure. The implemented model was evaluated in terms of its effectiveness in justifying XML rewriting attacks [11][12]. Generally, Service Oriented Architecture systems are susceptible to in transit sniffing or spoofing. XML rewriting attack refers to the class of attacks, which involve modifying the SOAP messages.

We focused in three scenarios and did different types of XML rewriting attacks.

1) We generated a basic XML replay attack in which the new message was entirely replaced by the old message captured by our attack tool.

2) Performing a XML message replay attack when there are security headers present in the web service messages.

3) Performing XML message redirect attack when there are security headers present in the web service message.

Figure 6. Attack Scenario Setting.

3.2 Performance Measurement in Multi Tenant Cloud Server

Multitenancy was used to study the impact of migration of the proposed end to end security solution to the cloud. In order to ensure that different services were deployed on different physical machines, large machine object were launched in different availability zones of multi tenancy in the East region as seen in the figures below. Figure below reports the average response times for the first 400 request to the Evacuation Timer Service for the baseline and taint analysis cases. As seen in the graph.
Figure 7 Baseline Experiment Setup in the Multi tenant Cloud Server

Figure 8. Taint Analysis Experiment Setup in the Multi tenant Cloud Server

Figure 9 reports the average response times for the first 400 request to the Evacuation Timer Service for the baseline and taint analysis cases. In this graph the response time are still very close up to 4 requests. The overhead is larger but acceptable for 8 and 16 requests.
We also conducted experiments to calculate the performance of the trust broker under growing loads for the session feedback method. A large machine instance in the Multi-tenant cloud server East region was used to host the trust broker for these experiments. In Figure 10, we shows that the rate of requests was kept fixed by setting the delay between consecutive requests by a single thread to 100 milliseconds and increasing the delay proportional to the number of simultaneous threads. The results for these experiments show that the Trust Broker is able to handle 64 simultaneous requests in around 150 milliseconds and 128 requests in around 200 milliseconds.

In the second set of experiments, bursts of requests were sent at increasing rates, i.e. the delay between the consecutive requests of all client threads was set to 100 milliseconds. The results of these experiments show that the increase in the rate of requests causes a small overhead in the response time up to 64 client threads; however there is a big jump in the overhead after 128 client threads, at which point load balancing should be considered.
IV. DISCUSSION AND FUTURE WORK

4.1 Security Procedure Enforcement:

The main goal of this paper has been to design a security auditing architecture. So it takes a retroactive approach for external service calls and only reports the external invocation events back to the TB. However, it can easily be converted into a proactive mechanism to enforce client’s policy. This could be realized by adding a policy engine XACML [14] to TB and employing TA module proactively. Another variation could be sending the upcoming service invocation to the TB and requiring services in an invocation chain to get a confirmation for the next service they will invoke.

Extension of Cloud Computing: The proposed architecture partly mitigates the threats posed by multi-tenancy too, as in the case of a certified service being under attack, the TA module deployed on the server will report malicious behavior to the TB. Even in the case of a TA module under attack, it will be possible to detect that attack with a slight modification in the architecture. For that, the TB would need to wait on feedback from the TA module of every certified service, which is known to be invoked by a previous service and update the session history for that service call with a warning after a timeout period during which no feedback is received. Investigating new threats for SOA-based systems in cloud computing environments: Deploying services in the Cloud brings up the question of potential security threats due to multi-tenancy. In future work, we will investigate the possible effects of multi-tenancy on the proper functioning of the proposed solution with session history parameter. Experiments will be performed with multi-tenant Cloud servers, where attacks will be targeted from one virtual machine to another to disrupt the functioning of the TA component and solutions to the problems will be investigated.

V. RELATED WORK

Many researchers have studied security of service-oriented architectures. We address the security issues in SOA by focusing on web service standards [13][14]. We identify the complexity of certifying SOA services due to the difficulty in representation of security controls in web services standards specifications in a constant manner for verification. In [15] and [16] the identification of trusted services and dynamic trust assessment in SOA are studied. We introduce a framework called RATEWeb for trust-based service selection and work based on peer feedback. It is based on a set of decentralized techniques for evaluating reputation-based trust with ratings from peers. However they do not take into account initial service invocations and the secondary services in compositions.

Approaches like [15] and [16] are not suitable for SOAs with a lot of services because the monitoring system would need to collect demanding information from a lot of peers and consumers, which would make it very expensive. Generally, taint analysis has been a low level mechanism which has been used for binary program analysis [19]. But on the other hand, low level taint analysis mechanisms lead to a considerable overhead which is not suitable for real world services. Moreover, they are dependent on specific hardware architectures which are not suitable for real world deployment. DIFC (Decentralized Information Flow Control) has been an active area of research in the past few years. Researchers in [18] [19] [20] have proposed different labeling mechanisms to secure applications from untrusted codes. Their approach needs a complete redesign of the OS which is not practical in the federated SOA settings. To overcome this problem,
we [21] propose a language level solution for information flow control which assigns labels to every program object that incurs a substantial overhead. In both mechanisms, we have to change the source codes of the services. Therefore, we lose transparency which is a key factor in implementation of a technology by industry.

VI. CONCLUSION
In this project we proposed an end-to-end security solution for SOA, which is based on the introduction of two new security components, i.e. the “Taint Analysis” module and the “Trust Broker” service. By providing the ability to track external service invocations in the completion of a service request and maintaining dynamic trust values for services, the proposed architecture allows clients to be informed about the full chain of service invocations in a request and possible misbehavior by services involved in the request. This architecture both makes it possible to judge the quality of the response received by the client and increase the chances of selecting trustworthy services using the reputation based system. Although the security architecture described above seems to take more of a proactive approach for external service calls, it can easily be converted into a proactive one by either the prevention of external service calls to untrusted services by the TA or by requiring services in an invocation chain to contact the TB to get confirmation for the next service they will invoke. The latter approach will introduce additional delays in the response for the clients, but may be preferred for preventive security.

The proposed end-to-end security architecture is fully compatible with common Web services standards, as the services and data communication protocol are not affected by the security related modifications in the general SOA structure. The minimal set of web services standards basic to overcome the security challenges along with the proposed security components TA and TB were identified as WS-Security to ensure client and service authenticity as well as message level security through encryption and signing; and WS-Trust for the generation of security tokens required for authentication. By securing the communication between the taint analysis modules and the trust broker using Web Service-Security, the proposed system ensures authenticity of session feedbacks, hence preventing unfair increase/decrease of trust values of services due to targeted feedback from malicious parties.

Experiments performed in the multi tenant cloud server suggest that the proposed solution causes small overhead in terms of the service response time up to a certain load on the server, at which point load distribution should be considered. The same argument holds for the Trust Broker service as well; i.e. to avoid being a single point of failure prone to denial of service attacks, the TB should distribute its load over multiple servers. This makes the Cloud the best option for hosting the TB service. With elastic load balancing achieved by on-the-fly allocation of resources and creation of virtual machines, a TB service in the Cloud will be able to meet the demands for different service request loads and prevent waste of resources in the case of decreased service traffic.

REFERENCES


