# Security and Access Control Representation for Generating Valid Composite Policies

[1]B. Muruganantham, [2]GS Anurag Rao

[1]Assistant Professor (Sr.G), [2]M.Tech
Computer Science and Engineering, SRM University
[1]murganantham.b@ktr.srmuniv.ac.in, [2]anuragrao17@gmail.com

***Abstract* - The web services contain atomic and composite services, the composite service able to invoke other composite or atomic service. The problem here is while composite service invokes other services, the policy agreements and type of security used should be matched with the invoked web services. As a result there will inconsistency among the web services and also there will be lack of message protection policy. To solve this problem, developers have to define policy of composite service by hand by referring to the policies of the invoked web services in the composite services. However it is hard very hard to complete a policy composition without any inconsistencies, because the process definition and security policies are complex and it is not clear how to compose policies to maintain consistency. So a proper policy composition engine which can eradicate the inconsistency between web services is needed. Hence in this project, a policy composition engine which is able to verify the polices between web services that has different type of access control policy and message protection policy compared to composite web service is proposed. The policy composition engine adjusts the policy of composite services based on web services invoked. Meanwhile a revised composite policy might have redundant requirements. In order to avoid this problem, the redundant representations are merged or removed in a supervised manner that examines the definitions of all implemented services.**

***Keywords* - Web service, Atomic web service, Composite Web service, Security policy**

## I. INTRODUCTION

Different organizations use many software systems for management. Software systems often need to exchange data with other system and a web service is a procedure of communication that allows two software systems to exchange their data over the internet. The system that requests for the data is called a service requester.Whereas the software system that would follow the procedure to serve the request and provide the data is called a service provider with the help of SOAP protocol. The service provider Checks the service request and sends structured data in an XML file, using the SOAP protocol. This XML file would be validated again by the service requester using an XSD file.

Different software might be built using different programming languages, and hence there is a need for a procedure for data exchange that would not depend on any programming language .Almost all types of software's can however interpret XML tags. So the web service can use XML files for data interchange. Rules for communication need to be established.

- How the data can be requested from the other system.
- Which specific variables are needed for data request.
- What would be the structure of the data produced. Normally, data is interchanged in XML files and the structure of the XML file is validates against an .xsd file
- How the error messages need to be displayed when certain rule for communication is not observed , to make troubleshooting easier.

All of these rules for communication are defined in a file called WSDL (web service description language), which has the extension .wsdl. The service provider sends a WSDL file to UDDI. Then the service requester contacts UDDI to find out who is the provider for the data it needs, and then it contacts the service provider using SOAP protocol. The service provider checks the service request and sends structured data in an XML file, using SOAP protocol. This XML file would be validated again by the service requester using an XSD file.

A directory called UDDI gives u exact information in such a way that which Software system should be contacted for which type of data so when one of the software systems needs certain particular data, it would contact UDDI and find out at the other end that which system should it contact for receiving that data. Once the software system finds out at the other end which system it should contact ,it would establish the connection with the system using SOAP(Simple Object Access Protocol).The service Provider System Initially validate the data request by referring to the WSDL file and process the request and send the data using SOAP protocol.

### *Atomic Web Service*

It is a web service, whose implementation is itself contained and does not invoke any other services.

### ***Composite Web service***

If the execution of a web service's business logic involves the invocation of other web services, it is necessary to combine the functionality of several web services.

***Difference between Atomic service and Composite service***

A composite service is a service whose implementation calls other service. When it comes to atomic service, Its implementation is self contained and does not invoke any other services. A composite service can act as both Service Provider of the (composite) service and as a service consumer of its child services. The composite service can be considered as the summation of the child services into a bigger service.

***Security Policy***

Security is established by creating a security policy. It is a set of laws controls the security system information and provide him with the levels of protection trusted. This contains written guidelines, which outline what steps to take and what procedure to follow in the pursuit of security. A policy is typically a document that outlines specific requirements or rules that must be met.

***Service Oriented Architecture***

An application based on the Service-Oriented Architecture (SOA) consists of a group of services, which is considered as a composite service. A composite service can be invoked from other composite services, and hence the application would have a repetitive structure. Securing the SOA application is an important nonfunctional requirement. However, defining a security policy for a composite service is not easy because the policy should be consistent with the policies of the external services invoked in the composite process. Therefore in this paper we propose a security policy composition mechanism that uses in this project, a policy composition engine which is able to verify policies between web services that has different type of access control policy and message protection policy compared to composite web service is proposed .The policy composition engine adjusts the policy of composite services based on web services invoked. Meanwhile a revised composite policy might have redundant requirements. In order to avoid this problem, the redundant representations are merged or removed in a supervised manner that examines the definitions of all implemented services.

## II. EXISTING SYSTEM

In the existing system there is way for satisfying only functional requirements and there is no proper implementation for nonfunctional requirements such as security, and rules should be defined by the users so it involves many complexities. In order to achieve the security of the web service there should be a proper security rules should be present, but there are no clear rules are specified for security policy, and there is no clear rule for composing atomic and composite web service without inconsistency.

***Issues in Existing System***

- The existing system uses a meta-data policy describing how it should be composed with other such meta-data policies but it acquires complexity and more cost.
- There is a static approach for defining the composition policies, so it results not a proper consistent rule.
- The existing system applies a predicate logic work flow management, the security policy focus is access control for a specific workflow but the transformation from work flow into logic is not provided.

## III. RELATED WORKS

- A.J. Lee at al [1] proposed the composite security policies and they apply the concept of logical defeasible events to test the security policies written in WS-Security policy.
- Huangqin jiang and hongqi zhang has proposed a composite web service access control model combining the idea of attribute based access control. This model can be composed according to the relations of services composition and access control polices of component web services and also this model supports access control for services parameters, single web services and composite web services [2].
- Fumiko satoh and takehiro Tokuda has proposed a Security policy composition mechanism from existing polices of external services. They have created a security policy of a composite service automatically based on a predicate logic with support for two approches of policy composition. And also they focused on three types of security policies such as data protection policy access control policy and composite process policy [3].
- Daisy daiqin he and Jian yang has mainly focused on security policies issues in cross-organization collaboration in the context of web services. They have identified different collaborations types, analyzed relationships between policies, checked compatibility for collaboration and defined rules for generating integrated security policies. They also proposed a framework for handling authorization control for business collaboration [4].

## IV. PROPOSED SYSTEM

The aim of this proposed system is to provide consistency between atomic web service and composite web service by implementing policy composition engine.

- By implementing a proper policy composition engine, there will no user involved in middle, in order to perform composition policy checking for inconsistency [1].
- The proposed addresses two approaches top down and bottom up approach, by implementing this approach it provides solution hints in order to resolve policy inconsistency.
- While checking for the inconsistency between atomic web service and a composite service there might be a chance for redundancy, so the proposed system also encounters the redundant variables in composite service and tries to eliminate those redundant variables. The architecture framework of our proposal is depicted in fig 1.

- The main elements of our framework include atomic web service, composite web service, policy composition engine, inconsistent variable [5] [6].
- First user request submitted to the request manager which processes the user request and generates a comprehensive message which it further submits to the service repository. The service repository contains atomic web services. According to the user request the significant atomic web services policies combine to build a composite policies .The policy composite engine is designed to verify the inconsistencies between policies of composite and atomic web services. The inconsistency is identified if the flag is set to 'true'. While checking for the inconsistency between atomic web service and a composite service there might be a chance for redundancy, so the proposed system also encounters the redundant variables in composite service and tries to eliminate those redundant variables. The policy composition engine identifies & eliminates the redundancy.

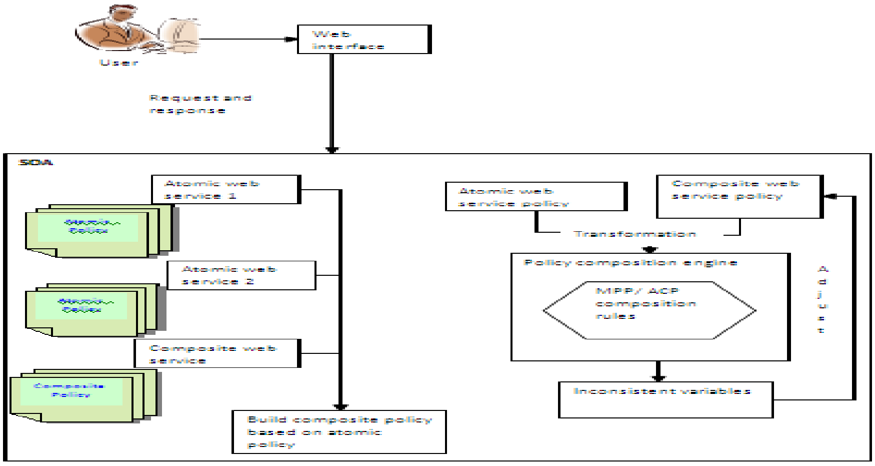

Figure 1

**Policy composition engine consists of two policies:**

**Message Protection Policies (MPP)**

- A Message might travel between various intermediatories before it reaches its destination. Therefore MPP is important to protect (integrity & confidentiality) at point to point transport level.
- The Message protection policy represent message protection such as confidentiality, integrity during the message interchange.
- Creation of MPP Composition Rule used by first order predicate logic. Here, we represent rules in a Prolog format and the uppercase letters show the types of the variables.
  Ǝ var1, var2, prop1, prop2
  Assign (var1: Variable, var2: Variable)
  varProp(var1, prop1:Property)
  varProp(var2, prop2:Prperty),
  Consistent (prop1, prop2).
  assign: A variable var1 is assigned to to variable var2.
  varProp(var, prop):A variable var has a Propety prop.
  Consistent: The two properties prop1 and prop2 are consistent.

**Access Control Policies (ACP)**

- Authorization is needed in order to control access to resources. Once authenticated, authorization mechanisms control the requester access to appropriate resources. Policy determines the access rights of a requester.
- The user request satisfies the message protection policies and access control policies it provides the significant service to the user. Otherwise it doesn't provide service to the user.
  Ǝ var1, var2, ope1, ope2, oprop1, oprop2
  assign(var1:Variable, var2:Variable)
  operation(ope1:Operation, var1),
  operation(ope2:Operation, var2),
  opeProp(ope1, oprop1:OpePrpoerty),
  opeProp(ope2, oprop2:OpeProperty),
  consistent(oprop1, oprop2).
  operation(ope, var):The operation ope uses a variable var.
  opeProp:A property of an operation.
  Consistent:It returns true if the properties of two operations, oprop1 and oprop2, are consistent.

**Policy composition method**

In this paper we propose a method for composing and validating polices in composite services according to the rules that depend on the variable assignments and their properties and also we adjust the policies if any inconsistencies arises when one service invoke other service i.e. when a composite service invoke a atomic service if any inconsistency arises with the message protection policy attached to these services then those inconsistencies are resolved by the policy composition engine. Suppose if the security policy of the travel reservation requests a customer ID and security policy requires that the request message should be signed and encrypted by a high level algorithm. So hence we can confirm that the customer id itself is highly confidential data, so if any other service that uses this customer id should maintain the high security by using the same high level algorithm. Another Scenario when two variables are regarded as identical they should have the same security properties .This is the basic idea and behind the policy composition rule and is one of the contribution of this paper.

**Example:**
If there are three services of them two are atomic services and one is composite service
Atomic service1-Flight Booking A, Flight Booking B
Atomic service2-Hotel booking A, Hotel Booking B
Composite service-Travel agency
Flight Booking-Here the user would be able to book the flight with the help of this service.
Hotel Booking-Here the user would be able to book the Hotel with the help of this service.
Travel Agency-The travel Agency would provide both the services of Hotel and Flight Booking.
The algorithms which are used for Message Encryption and Signature are

- SH1-Secure Hash Algorithm
- RSA-RSA Encryption Algorithm
- MD5-Message Digest Algorithm
- AES-Advanced Encryption Standard
- Signature Algorithms-SH1 with RSA,MD5 with RSA
- Encryption Algorithm-AES,RSA/ECB/PKCS1padding

| | Access Control Policy | Message Protection Policy | Travel Agency and policy Composition Engine |
|---|---|---|---|
| Flight Booking A | Authorized User | SH1 with RSA | Any Inconsistencies regarding MPP are resolved |
| Flight Booking B | Authorized User | MD5 with RSA | Any Inconsistencies regarding MPP are resolved |
| Hotel Booking A | Authorized User | AES(Encryption Algorithm) | Any Inconsistencies regarding MPP are resolved |
| Hotel Booking B | Authorized User | RSA(Encryption Algorithm) | Any Inconsistencies regarding MPP are resolved |

Here the travel agency consists of the file called Policy watch dog which in take cares of in case if any of the service changes its encryption or signature algorithm theopposite service which is invoking these service would be intimated and adjusted according to the invoked service message protection policy. The policy composition engine would Resolve any Message protection policy inconsistencies which would arise in the when one service invoke other service.

## V. CONCLUSION

An application based on the Service-Oriented Architecture (SOA) consists of an assembly of services, which is referred to as a composite service. A composite service can be implemented from other composite services, and hence, the application could have a recursive structure. Securing an SOA application is an important nonfunctional requirement. However, specifying a security policy for a composite service is not easy because the policy should be consistent with the policies of the external services invoked in the composite process. Therefore, this paper proposes a security policy composition mechanism that uses in this project, a policy composition engine which is able to verify policies between web services that has different type of access control policy and message protection policy compared to composite web service is proposed .The policy composition engine adjusts the policy of composite services based on web services invoked. Meanwhile a revised composite policy might have redundant requirements. In order to avoid this problem, the redundant representations are merged or removed in a supervised manner that examines the definitions of all implemented services.

## REFERENCES

[1] A.J. Lee, J.P. Boyer, L.E. Olson, and C.A. Gunter, “Defeasible Security Policy Composition for Web Services,” Proc. Fourth ACM Workshop Formal Methods in Security (FMSE ’06), pp. 45-54, 2006.

[2] Huangqin Jiang, Hongqi Zhang “Access Control Model for Composite Web Services” ICCT International Conference in Communication Technology, 2012.

[3] Fumiko Satoh and Takehiro Tokuda “Security Policy Composition for Composite Services” ICWE International Conference in Web Engineering, 2008.

[4] Daisy Daiqin He and Jian Yang "Security Policy Specification and Integration in Business Collaboration" IEEE international Conference on Service Computing, 2007.

[5] K. Bhargavan, C. Fournet, and A.D. Gordon, "Verifying Policy- Based Security for Web Services," Proc. 11th ACM Conf. Computer and Comm. Security,pp. 268-277, 1992.

[6] Y.H. Li, H. Paik, B. Benatallah, and S. Benbernou, "Formal Consistency Verification between BPEL Process and Privacy Policy," Proc. Int'l Conf. Privacy Security and Trust Conf.: Bridge the Gap between PST Technologies and Business Services (PST '06), 2006.