A Context-Based Security Framework for Cloud Services

Hassan Saad Alqahtani
University Campus Milton Keynes
Milton Keynes, UK
Hassan.alqahtani@beds.ac.uk

Ghita Kouadri Mostefaoui
University College London
London, UK
g.kouadri@ucl.ac.uk

Zakaria Maamar
Zayed University
Abu Dhabi, UAE
Zakaria.maamar@zu.ac.ae

ABSTRACT
This paper discusses the use of Aspect-Oriented Programming (AOP) as an efficient way to handle cloud computing frontend security concerns. Without AOP, the necessary security code would be mixed with the business logic that the cloud service provider implements. This makes the maintenance of both code and business logic tedious and prone to errors. The proposed aspect-oriented approach in this paper is built upon a Web services frontend to the cloud service. Three types of context are taken into account when tuning the aspects (security services). The contexts contain various details on the environment of the Cloud and the Web services, which permit activating the necessary aspects in response to these details. A set of experiments validating this approach, are also reported in this paper.

Keywords
Cloud computing, frontend, Web services, aspect-oriented-programming, security, cross-cutting concern.

1. INTRODUCTION
Cloud computing is an internet-based framework for enabling a pool of shared resources/capabilities on demand [1][2]. NIST [3] defines five essential characteristics for the cloud computing service, which are resource pooling, measured service, on demand services, elasticity and ubiquitous access. Other technologies are used to support the cloud computing paradigm; such as, web services and virtualization. Practically, the internet and web services will be used in order to deliver the cloud service to the customer. The web services play an important role in cloud computing technology, beside their capabilities to offer fast and cheap access to the provided resources/capabilities; Web services offer system interfaces that could be used for consuming and managing the delivered cloud services [4]. In addition to sharing business logic, data and processes through a programmatic interface across a network; Web services allow different applications from different sources to communicate with each other, Web services are not dependent to any operating system or programming language; for that, combining Web services and cloud computing allow developing unified friendly interface for end-users in an interoperable way [5].

Web services frontend represents a gate for customers to interact with the cloud service [4]. Web services are hailed for their capacities in developing loosely coupled business processes that can spread over organizations' boundaries [6]. In the last years there is a number of efforts that have been put into supporting the acceptance of Web services among the IT community by making them the technology of choice when developing such processes. However, several still have doubts about Web services' capacities when it comes to addressing critical concerns like security [7] and fault tolerance [8]. In cloud computing essential security services such as authentication are implemented at the frontend. To implement this frontend, the majority of cloud services providers rely on web services [9].

For a proper handling of security we advocate for a clear separation between the business logic that underpins a Web service operation and the security requirements that restrict this operation. This handling is rendered possible using Aspect-Oriented Programming (AOP) [10]. AOP allows confining a concern code into one single module without making it scatter over other modules of a system. This helps minimize maintenance efforts in the case of any new requirement arises. Examples of initiatives adopting AOP for Web services include [11].

Our proposed approach to handle security concerns is built upon three levels referred to as resource, component, and composite. The status of each level in terms of actions taken and techniques adopted is reported in a specific structure known as context. In this approach these concerns (security services) are authentication and access control; which ensure that each user will have the assigned privileges only, and avoid un-authorized access attempt. On top of the three contexts, a dedicated context associated with security is also defined. It is a state of the working environment that requires taking one or more actions.

The rest of this paper is organized as follow. In the next section the security-based, aspect-oriented approach is presented in terms of architecture and security aspects configuration. Section 3 depicts our proof-of-concept. Related work is discussed in Section 4. Finally, we draw a set of conclusions in Section 5.

2. OUR APPROACH

2.1 General architecture
Context has proven to be necessary to take into account when tuning the operation of Web services [12]. The context defining Web services operation refers to the users of the Web services (e.g., stationary versus on the move), level of expertise (e.g., expert versus novice), computing resources (e.g., fixed versus mobile), time of the day (e.g., in the afternoon versus in the morning), to mention just a few.

Figure 1 presents the way we handle aspects in the proposed approach. Three levels of abstraction exist: user, Web service, and cloud resource. The constituents of each level are tracked using specific contexts, for instance U-context, W-context, and R-context. The connection between user, Web service, and cloud resources levels is implemented in "invokes" and "operates upon" format. The proposed approach has a number of features; for instance, separating each level concern through aspects, and tracking the security requirements of the delivered cloud services, contextually.

ICCASA 2014, October 15-16, Dubai, United Arab Emirates

DOI 10.4108/icst.iccasa.2014.258610

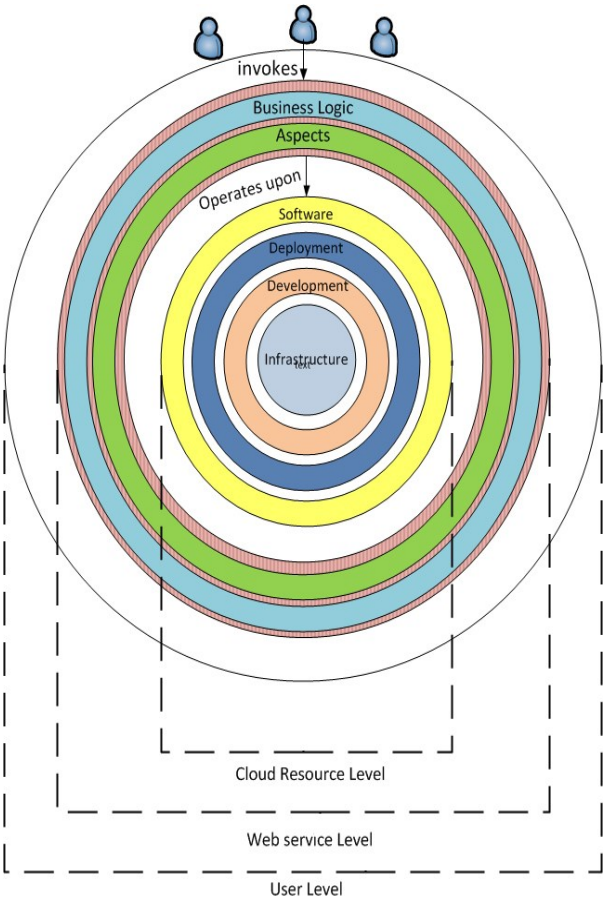

Figure 1. Overview of the overall architecture

The Web service level refers to the context-aware Web service. A Web service could be divided into two main segments, which are the business logic and the aspects. On the one hand the business-logic represents the actions that the Web service implements as part of the delivered functionality (e.g., query data). On the other hand, the aspect part represents the non-functional needs, for instance, the cross-cutting concerns that influence on the taken actions and interactions of the cloud service such as security and logging.

Figure 2 presents an example of how the proposed approach works, where the cloud provides storage service to their customers. The service provider offer three types of customers as follow:

- Data Owner: creates/edits the document, and grants/revokes the permission from other user that associated with the Data Owner documents.
- Editor: reads/writes the Data Owner documents.
- Reader: reads the Data Owner document only.

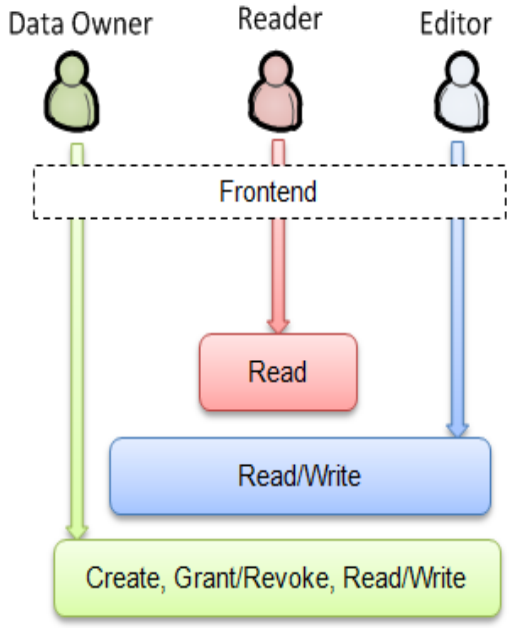

Figure 2. Overview of the overall architecture

The cloud resource level is about context-aware resources. The cloud resources refer to the delivered cloud resources/capabilities. The received requests of Web services will be scheduled in order to execute by prioritizing these requests when the required resources/capabilities are occupied by other requests.

2.2 Configuring security aspects

Figure 3 illustrates the operation of the approach. This operation consists of a set of aspects that have been selected in order to offer the necessary security aspects that could help protecting the environment of Web services (these aspects can be shown in Figure 3). Practically, the aspects selection process integrates context and policies. The former offers information of the environment that includes users, Web services, and resources; and the policies will activate the proper security aspects according to the provided information. Beside the W/R/U-contexts (refer to Figure 3 for context description) we suggest a, security context (S-context in Figure 3). It is received the details from W/R/U-contexts and is involved when activating the policies for weaving active security aspects. According to Kouadri Mostefaoui [13], “a security context is a state of the working environment that requires taking one or more security actions”. In the following and relying on a previous work in [14], we overview some arguments that populate each type of context and illustrate the specification of a policy in Java language code as a first approach.

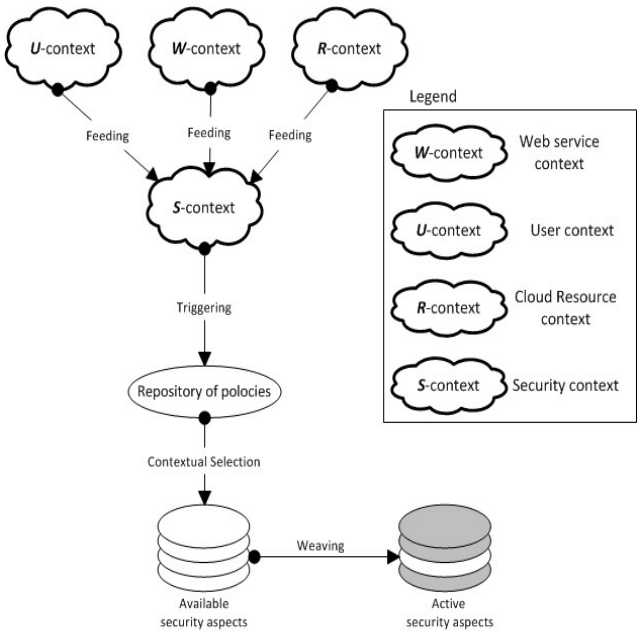

Figure 3. The three types of contexts and the security aspects

There are a number of arguments in W-context ; such as, signature (in order to identify the source of the exchanged messages), security mechanism (defines the used mechanism for authenticating the clients of the delivered cloud service), security status (shows the result of authentication checking procedure), and violation (shows the type of security breach that the message was subject to). The arguments in U-context are user's identity, role, etc. Some arguments in R-context are: the availability (distinguishes if the cloud system resources/capabilities are able to accept more connections or not), and the violation (presents the type of the security breach that the cloud service is engaged in). Additionally, the arguments in S-context are: the security violation per Web service/Resource, security status per Web service/ Resource, , and authentication method per Web service/Resource.at the end, the main function of S-context point out which authentication mechanisms (username/password pairs, binary certificate, etc.), certificate algorithms, etc. are supported by all components and when they are active.

3. PROOF OF CONCEPT

3.1 Case study

The used case study is a typical IaaS (Infrastructure as a Service) model. Figure 4 shows the architecture of the system, which consists of four elements as follow:

- User: there are three level of users, which are customer, customer service (deployment and direct interact with customer) and developer (development and interact with the cloud infrastructure).
- Frontend: This will be applied through the Web service to grant access.
- Development platform: where the delivered software will be developed, tested, deployed and managed.
- Cloud infrastructure: contains the shared resources/capabilities.

The Web services provide a way to remote access, consume, develop and manage the provided resources/capabilities.

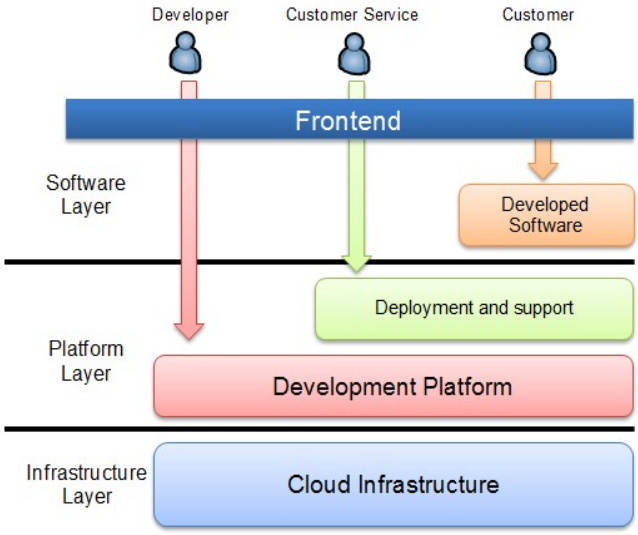

Figure 4. System architecture of the delivered cloud service

In the current case study, the authentication method that needs to be applied on users is different depending not only on their role within the cloud system (developer, customer service, general public users) but also on the type of the query (deploy, update, download files, test, etc.). Those authentication methods range from CAS (Central Authentication Service), database authentication or anonymous user/password method. This is the first identified cross-cutting concern.

Another cross-cutting concern in the current case study is logging users' activity. For reporting purpose, requests to the Web services are logged in a statistics database. Also, a periodical, check is performed by an automatic process (linux cron job) on the Web services to ensure they are available and working as expected. Those frequent accesses are not stored as they will clutter the statistics database. Only real requests (by human users) are logged. It is therefore needed to identify between the two types of requests.

3.2 Development tools

The current prototype is implemented using the following technologies:

- ***Apache Axis2/Java:*** It's an open source Web services framework.
- ***Apache Tomcat:*** It is an open source Web server and servlet container. It includes tools for configuration and management, but can also be configured by editing XML configuration files. Web services implemented using axis2 are deployed in Tomcat.
- ***Spring AOP + AspectJ:*** Both are implementations of the AOP (Aspect-Oriented Programming) paradigm. A comparison between the two implementations can be found here [15]. In order to have the best of both worlds, we use a combination of Spring AOP and AspectJ for implementing security aspects.

3.3 Experiments

3.3.1 Policy specification

In order to illustrate our prototype, we will focus the following simple context-based policy. The U-Context represents the user context. It may take three different values; developer, customer service and customer. The latter correspond to the different roles of the requesting Web service client. The W-Context represents the Web service context. For simplicity reasons we retain two cases; ready and busy representing the states of the service and this can be computed by checking the number of simultaneous requests the Web service is handling at the time. The R-Context represents the cloud resource context. In our case, the resources/capabilities could be divided into two layers: the first layer contains the developed software and second layer consists of the development platform and the cloud infrastructure. Finally, the S-Context, which denotes the security context of the running system is the combination of the different states of the three former contexts, namely U-Context, W-Context and R-Context (Table 1).

Table 1. COMPOSITION OF THE SECURITY CONTEXT

U-Context:developer, customer service and customer *W-Context*: ready, busy *R-Context*: accept_requests, refuse_requests *S-Context*: (*U-Context, W-Context, R-Context*)
Composition of the security context

Based on the three types of context discussed earlier, in the following we present the specification of three sample security policies (Table 2.).

Table 2. CONTEXT-BASED SECURITY POLICIES SPECIFICATION

IF U-Context= developer OR Customer and W-Context.ready AND R-Context = accept_requests *CASAuth_aspect* *StoreStats_aspect* *IF U-Context= tester AND W-Context.ready = true AND R-Context = accept_requests* *DatabaseAuth_aspect* *IF W-Context= busy OR R-Context = refuse_requests* *DelayRequest_aspect*
Context-based security policies specification

3.3.2 Web service implementation

The frontend will provide direct access to both (1) the developed software and (2) the development platform and infrastructure.

Figure 5 illustrates a portion of the WSDL (Web Service Definition Language) file. There are a couple of other utility methods *echoString* that returns a string based on the client request and that is useful when testing whether the Web service is alive without the need to authenticate. *getVersion* is another utility method that returns the version of the running Web service. The core method is actually *queryData* which is called by a client to read data stored in the cloud

```
<wsdl2:interface name="ServiceInterface">
   <wsdl2:fault name="BadDataException" element="tns:BadDataException"/>

   <wsdl2:operation name="echoString" pattern="http://www.w3.org/ns/wsdl/in-out"
      wsdlx:safe="false">
      <wsdl2:input element="tns:echoString" wsaw:Action="urn:echoString"/>
      <wsdl2:output element="tns:echoStringResponse" wsaw:Action="urn:echoStringResponse"/>
      <wsdl2:outfault ref="tns:BadDataException"     wsaw:Action="urn:echoStringBadDataException"/>
   </wsdl2:operation>

   <wsdl2:operation name="getVersion" pattern="http://www.w3.org/ns/wsdl/in-out"
      wsdlx:safe="false">
      <wsdl2:input element="tns:getVersion" wsaw:Action="urn:getVersion"/>
      <wsdl2:output element="tns:getVersionResponse" wsaw:Action="urn:getVersionResponse"/>
      <wsdl2:outfault ref="tns:BadDataException"     wsaw:Action="urn:getVersionBadDataException"/>
   </wsdl2:operation>

   <wsdl2:operation name="queryData" pattern="http://www.w3.org/ns/wsdl/in-out"
      wsdlx:safe="false">
      <wsdl2:input element="tns:queryData" wsaw:Action="urn:queryData"/>
      <wsdl2:output element="tns:queryDataResponse" wsaw:Action="urn:queryDataResponse"/>
      <wsdl2:outfault ref="tns:BadDataException"     wsaw:Action="urn:queryDataBadDataException"/>
   </wsdl2:operation>

</wsdl2:interface>
```

Figure 5. Web service WSDL

The schema file in Figure 5 details the input and output of the *queryData* method. It takes as input the credentials (username and password) of the user along with his/her query string and returns a complex object containing the results of the query from the cloud service, plus a status message (*Ok, Fail*) and a message detailing how the processing of the query request went.

The client user doesn't need to write a SQL query as input, but will rather build an object with the names of the columns he/she wants to query, along with the constraints (range of query) and the Web service will automatically build the equivalent SQL query based on the backend cloud database schema.

The output response is a XML string containing SQL-like elements including the number of columns returned by the SQL query, the names of the columns, the number of records returned (*numberRows*) and the actual data i.e. a set of rows.

```
<xs:element name="queryData">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="credentials" type="aof4ws:credentials"/>
         <xs:element name="queryString" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="queryDataResponse">
    <xs:complexType>
      <xs:sequence>
         <xs:element name="numberColumns" type="xs:int"/>
         <xs:element name="columnNames" minOccurs="0" maxOccurs="unbounded" type="xs:string"/>
         <xs:element name="numberRows" type="xs:int"/>
         <xs:element name="row" minOccurs="0" maxOccurs="unbounded" type="xs:string"/>
         <xs:element name="status" type="aof4ws:Confirmation"/>
         <xs:element minOccurs="0" name="message" nillable="true" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Figure 6. Web service schema file

Each request to the Web service method *queryData* will trigger the enforcement of the security policy. The following code snippet illustrates this functionality (Figure 7).

```
public uk.ac.aof4ws.www.services.scws.QueryDataResponse queryData(
        uk.ac.aof4ws.www.services.scws.QueryData queryData)
        throws BadDataExceptionException {

    // extract input data from request
    //...

    // trigger the policy manager
    PolicyManager manager = new PolicyManager();
    manager.triggerPolicy();

    // return result
    //...

}
```

Figure 7. Web service *queryData* method implementation

3.3.3 *Context-based policy enforcement*

As stated in previous sections, the core context our system relies on is a security context that is composed of information gathered from the client, the Web service and the cloud resources. The PolicyManager class is responsible for identifying the current security context by computing the three different contexts, namely U-Context, W-Context and R-Context and then applying the needed checks in order to trigger the corresponding S-Context. In our application, the security context is implemented as a Spring bean e.g. SecurityContextBean (Figure 8).

```
public class PolicyManager {

    @SuppressWarnings("resource")
    public void triggerPolicy(){


        // identifying the current security context
        String uc = UContextManager.getContext();
        String rc = RContextManager.getContext();
        String wc = WContextManager.getContext();

        /*
         * Policy enforcement
         * */

        applicationContext appContext = new
                ClassPathXmlApplicationContext("Spring-AOPSecurity.xml");

        // security context A
        if ((uc.equalsIgnoreCase("developer")) || (uc.equalsIgnoreCase("customer"))
                && wc.equalsIgnoreCase("ready")
                && rc.equalsIgnoreCase("accept_requests")){

            SecurityContextA securityContexta =
                    (SecurityContextA) appContext.getBean("securityContextBeanA");
            securityContexta.getSecurityContext();
        }


        // security context B
        if (uc.equalsIgnoreCase("tester") && wc.equalsIgnoreCase("ready")
                && rc.equalsIgnoreCase("accepts_requests")){

            SecurityContextB securityContextb =
                    (SecurityContextB) appContext.getBean("securityContextBeanB");
            securityContextb.getSecurityContext();
        }
```

Figure 8. Context-based policy manager class

3.3.4 *Security aspects*

The following code snippet of the CASAuthAspect shows how the aspect intercept the execution of the SecurityContext defined earlier (Figure 9). This is done using AspectJ annotation @Before and @After. In the same manner the other cross-cutting concerns including database authentication and *delayRequest* are implemented. It is noted that the point is not how the functionality has been implemented but rather that an authentication aspect is available.

```
import org.aspectj.lang.JoinPoint;

@Aspect
public class CASAuthAspect {

    @Before("execution(* uk.ac.beds.cats.aopsecurity.SecurityContextA.getSecurityContext(..))")
    public void logBefore(JoinPoint joinPoint) {

        System.out.println("CASAuthAspect: logBefore() is running!");
        System.out.println("******");
    }

    @After("execution(* uk.ac.beds.cats.aopsecurity.SecurityContextA.getSecurityContext(..))")
    public void logAfter(JoinPoint joinPoint) {

        System.out.println("CASAuthAspect: logAfter() is running!");
        System.out.println("******");

    }
```

Figure 9. CAS authentication aspect class

3.3.5 *Running output*

For testing purposes, we set the following context information U-Context = “developer”, R-Context=”accept_requests” and W-Context=”ready” which leads to the identification of the first security context discussed in section ‘Policy specification’. The following shows the output of calling the queryData Web service method. We see that both aspects CASAuthAspect and StoreStatsAspects have been triggered as expected (Figure 10).

```
May 27, 2014 11:08:19 PM org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh
INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@6dd4ea6c: startup date
May 27, 2014 11:09:23 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
INFO: Loading XML bean definitions from class path resource [Spring-AOPSecurity.xml]
CASAuthAspect: logBefore() is running!
*****
StoreStatsAspect: logBefore() is running!
*****
getSecurityContext() is running!
*****
StoreStatsAspect: logAfter() is running!
*****
CASAuthAspect: logAfter() is running!
*****
```

Figure 10. Activated aspects when calling the queryData Web service method

4. RELATED WORK

In the current section, we focus on the security of the cloud system frontend, namely, webservices. We analyze the different contributions that have been proposed so far and that are related to our proposal.

The cloud and Web services have a lot of security threats and challenges that caused by dynamic and open environment. WS-Security is a Microsoft and IBM specification dedicated to Web services security [16]. It could be defended as a new standard for exchanging messages through Web services securely; For that, the WS-Security set how security tokens will be contained in the SOAP messages. Actually, the WS-Security is protractile to other security models; for instance, Public Key Infrastructure (PKI), Secure Sockets Layer (SSL), and Kerberos. Nowadays the majority of secure communication measures rely on the Transport Layer Security (TLS), that is used to ensure the security of interactions via making the communicated servers and clients collaborate in order to decide on the authentication process to adopt while they exchange the encrypted data. According to [17], TLS is not proper for the complex transactions like those involving Web services, and that is caused by the scalability limitations. In addition, SSL and the Virtual Private Network (VPN) cannot provide the required security for large number of requests that are expected through the Web services. The W3C’s Web services architecture adopts PKI to secure communications over public networks; But, the PKI’sinfrastructure complexity increases the deployment cost, processing time, etc. Additionally, PKI cannot be used for Web services intense interactions, because its cumbersome which will reduce the security of the provided services [11]. The eXtensible Access Control Markup Language (XACML) is an OASIS standard, that defines the access control decision service interface and the policy language, which defines the access control requirements. Practically, using request/response style for defining the used access controls will enable the query to ask whether or not the given action should be allowed: permit, deny, indeterminate, or not applicable. However, the AOP none of these techniques considers AOP to secure Web services.

Djordjevic et al. present an approach for securing Web service interactions via an integrated approach that combines hardware-based security mechanisms offered by Trusted Computing Platform [18]. In particular, the authors propose a trust-based architecture for protecting the enforcement middleware deployed at the policy enforcement endpoints of Web and grid services. The main motivation is to additionally secure execution environment of the applications, by providing virtual machine level separation that maps from logical domains imposed by Web services level enforcement policies. The approach presented in this paper, proposes the use of a context-based approach towards securing Web service interactions, using AOP to ensure separation of concerns between executing and securing Web services. The proposed architecture will offer a separation between the delivered application management procedures and the consumed infrastructure, which will be beneficial in the distributed scheme, in order to separate the responsibilities of each party.

A brief overview of interoperable security standards for Web services is presented in [19]. The key standards introduced and explained, are Security Assertion Markup Language (SAML), Extensible Access Control Markup Language (XACML), and several WS-* interoperability standards, such as WS-Trust, WS-Policy, WS-SecureConversation and WS-SecurityPolicy. The author also shows how these standards can be integrated into an overall Web services security architecture, and how the developer uses these standards to design the security approach. However, these standards lack the ability to specify separation of concerns between executing and securing Web services, as proposed in this paper. Importantly, the presented architecture might be proper for the security of cloud services that interact with heterogeneous modules, because of the ability of this architecture to offer a centric management.

Durbeck et al. present a security architecture for semantic Web services [20]. The key building blocks of this architecture are Web Service Modeling Ontology (WSMO) and Web Service Modeling Language (WSML), while the overall approach to secure semantic Web services is based on XACML. On similar lines, Rouached presents a security analysis for Web service compositions [21], and identifies the key issues, viz., setting and managing security policies, inter-organizational security issues and the implementation of high level business policies in a Web services environment. In particular, the management and maintenance of a large number of Web services needs appropriate authorization policies to be defined so as to realize reliable and secure Web Services. The required authorization policies can be quite complex, resulting in unintended conflicts, which could result in information leaks or prevent access to information needed. In that paper, Rouached discusses the authorization control for Web services compositions and proposes a logic based approach to ensure access control to such compositions. The AOP-based approach proposed in this paper to secure Web services can be enriched and improved upon the techniques presented in [20][21].

Mourad et al. present a framework for the dynamic enforcement of composite Web services security, which is based on integrating AOP concepts into BPEL [22]. This is achieved via a new

language called AspectBPEL, which is used to specify security policies as separate components, referred to as aspects, to be weaved systematically in a BPEL process. The injected aspects activate the security policies at runtime on specific join points. Some key features of this approach are: separation of business and security concerns of Web services; allowing update of security mechanisms of composite Web services at runtime; and modeling cross-cutting concerns between Web services in a modular fashion. A further enhancement of [22], which uses XACML to specify policies at join points in the BPEL process, is presented in [26]. Subsequently, BPEL flows with the needed security are generated into AspectBPEL security aspects to be weaved in the aforementioned process.

Goettelmann et al. propose an approach for deploying composite Web services on the cloud supporting security constraints, thereby ensuring sensitive data exchange [9]. Their approach uses partitioning techniques for fulfilling security requirements and optimizing communication costs. The partitions are deployed independently on different cloud platforms. Subsequently, these partitions depend on message exchange synchronization, as per the defined choreography on the cloud. Moreover, they also consider additional requirements related to data-dependencies and Quality of Service (QoS) disparities to optimize the execution of the outsourced composite Web service. However, since their approach does not consider separation of security concerns from Web service execution, it would become inflexible to implement in case its security requirements change.

Fernandez et al. discuss the proliferation of Web service security standards, and the overlap among them, which hinders their easy acceptance [24]. To address this problem, they propose to represent them as security patterns. Security patterns abstract the key aspects of a security mechanism and can thus be applied by non-experts. Their key aim is to enumerate security patterns based on already existing Web service security standards, show their usage, and also show how they relate to each other.

Ben Jemaa et al. analyze WS-SecurityPolicy and show that its lack of semantics hampers its use in matching security policies [25]. To resolve this problem, they present a semantic approach for specifying and matching Web service security policies. The approach consists in the transformation of WS-SecurityPolicy into an OWL-DL ontology and the definition of rules that automatically generate semantic relations that can exist between the provider and requestor security requirements. Zeng et al. present a policy-based architecture termed PBA4WSSP for Web services security processing [26]. In PBA4WSSP, a policy-based architecture is proposed to support multi-granular and customized security requirements. This architecture also provides the five security services, viz., integrity, confidentiality, non-repudiation, authentication and authorization.

Li [27] proposes an adaptive security model for communication on cloud computing, the core concept of the proposed approach is adapting the cloud security based on the current security status, by evaluation the cloud infrastructure performance and the resource consumption. This approach could be useful for these systems that have one security level only, because the developer has not considered the diversity in the security/access level.

The framework that proposed by Yu et al. [28] takes into account the diversity in the security levels; by providing two different levels of access control. However, the approach do not has the minimum amount of dynamicity to meet the diversity of security levels; it only offers two levels, which are not enough for the recent applications, regardless the way of provision (dynamic or static).

The developed approaches [29-31], take into account the instability of communication environment (Internet); by considering the heterogeneity and the internet vulnerabilities. Also, these approaches consider the diversity in security requirements, and offer dynamic management, but the developers do not consider the characteristic of cloud computing technology.

5. CONCLUSION

In this paper, we investigated the crucial issue of securing the cloud computing frontend through Web services using aspects. The AOP technique achieves a clear separation between functional and non-functional concerns in Web services. Our approach decouples security necessary code from Web service business logic, thereby providing easier adaptability and maintainability. In particular, our approach is contextual, comprising contexts at user, component and resource levels. These contexts enable the activation of the appropriate aspects in response to specific details related to Web service executions. We also presented a set of experiments validating our approach, via a proof of concept prototype. As future work, we plan to extend our approach by tackling other non-functional requirements such as self-healing, fault-tolerance and testing.

6. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin and I. Stoica, "A view of cloud computing," Commun ACM, vol. 53, pp. 50-58, 2010.

[2] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Comput.Syst., vol. 25, pp. 599-616, 2009.

[3] The NIST Definition of Cloud Computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf; Visited May 2014.

[4] L.M. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, "A break in the clouds: towards a cloud definition," ACM SIGCOMM Computer Communication Review, vol. 39, pp. 50-55, 2008.

[5] B. Furht, "Cloud computing fundamentals," in Handbook of cloud computing, Springer, 2010.

[6] M. Papazoglou, P. Traverso, S. Dustdar and F. Leymann, "Service-oriented computing: State of the art and research directions," Ieee Computer, vol. 40, pp. 64-71, 2007.

[7] Singhal, "Web Services Security: Techniques and Challenges," in Data and Applications Security XXII, Springer, 2008, pp. 158-158.

[8] G. Kouadri Mostéfaoui, Z. Maamar, N.C. Narendra and S. Sattanathan, "Decoupling security concerns in web services using aspects," in Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on, pp. 20-27, 2006.

[9] E. Goettelmann, W. Fdhila and C. Godart, "Partitioning and cloud deployment of composite web services under security

constraints," in Cloud Engineering (IC2E), 2013 IEEE International Conference on, pp. 193-200, 2013.

[10] S. Kleinschmager, Aspect-Oriented Programming Evaluated: A Study on the Impact that Aspect-Oriented Programming Can Have on Software Development Productivity, Diplomica Verlag, 2013.

[11] Schmidt, M. Beigl and H. Gellersen, "There is more to context than location," Comput.Graph., vol. 23, pp. 893-901, 1999.

[12] Z. Maamar, D. Benslimane and N.C. Narendra, "What can context do for web services?" Commun ACM, vol. 49, pp. 98-103, 2006.

[13] G. Kouadri Mostefaoui, Towards a Conceptual and Software Framework for Integrating Context-Based Security in Pervasive Environments, PhD thesis, University of Fribourg, 2004.

[14] N. Narendra and Z. Maamar, " Towards Context-based Tracking of Web Services Security", in Proceedings of The 7th International Conference on Information Integration and Web Based Applications & Services (iiWAS'2005), Kuala Lumpur, Malaysia, September 19-21, 2005.

[15] A. Gupta, "Comparative Analysis Between Spring AOP and AspectJ", available at: http://java.dzone.com/articles/comparative-analaysis-between, Visited August 2014.

[16] E. Zahoor, O. Perrin and C. Godart, "Disc-set: Handling temporal and security aspects in the web services composition," in Web Services (ECOWS), 2010 IEEE 8th European Conference on, pp. 51-58, 2010.

[17] Web Services Security. Version 1.1, February 2006, https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf, Visited May 2014.

[18] Djordjevic, S.K. Nair and T. Dimitrakos, "Virtualised Trusted Computing Platform for Adaptive Security Enforcement of Web Services Interactions," in Web Services, 2007. ICWS 2007. IEEE International Conference on, pp. 615-622, 2007.

[19] S. Lakshminarayanan, "Interoperable security standards for web services," IT Professional, vol. 12, pp. 42-47, 2010.

[20] S. Durbeck, C. Fritsch, G. Pernul and R. Schillinger, "A Semantic Security Architecture for Web Services The Access-eGov Solution," in Availability, Reliability, and Security, 2010. ARES'10 International Conference on, pp. 222-227, 2010.

[21] M. Rouached, "Security Analysis for Web Services Compositions," International Journal of Scientific and Engineering Research, IJSER, vol. 3, 2012.

[22] Mourad, S. Ayoubi, H. Yahyaoui and H. Otrok, "A novel aspect–oriented BPEL framework for the dynamic enforcement of web services security," International Journal of Web and Grid Services, vol. 8, pp. 361-385, 2012.

[23] S. Ayoubi, A. Mourad, H. Otrok and A. Shahin, "New XACML–AspectBPEL approach for composite web services security," International Journal of Web and Grid Services, vol. 9, pp. 127-145, 2013.

[24] E.B. Fernandez, O. Ajaj, I. Buckley, N. Delessy-Gassant, K. Hashizume and M.M. Larrondo-Petrie, "A survey of patterns for Web services Security and reliability standards," Future Internet, vol. 4, pp. 430-450, 2012.

[25] M. Ben Brahim, T. Chaari, M. Ben Jemaa and M. Jmaiel, "Semantic matching of web services security policies," in Risk and Security of Internet and Systems (CRiSIS), 2012 7th International Conference on, pp. 1-8, 2012.

[26] H. Zeng, D. Ma, Z. Li and Y. Zhao, "A Policy-Based Architecture for Web Services Security Processing," in e-Business Engineering (ICEBE), 2012 IEEE Ninth International Conference on, pp. 163-169, 2012.

[27] Wei Li, "An adaptive security model for communication on cloud," in Computer Science and Network Technology (ICCSNT), 2011 International Conference on, pp. 1964-1967, 2011.

[28] Shucheng Yu, Cong Wang, Kui Ren and Wenjing Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing," in INFOCOM, 2010 Proceedings IEEE, pp. 1-9, 2010.

[29] Seon-Ho Park, Jung-Ho Eom and Tai-Myoung Chung, "A Study on Access Control Model for Context-Aware Workflow," in INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on, pp. 1526-1531, 2009.

[30] S. Jha, N. Li, M. Tripunitara, Q. Wang and W.H. Winsborough, "Towards formal verification of role-based access control policies," Dependable and Secure Computing, IEEE Transactions on, vol. 5, pp. 242-255, 2008.

[31] Zhang Wendong and Zhang Kaiji, "A Role-Based Workflow Access Control Model," in Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on, pp. 1136-1139, 2009.