FEDERICA PACI, University of Trento MASSIMO MECELLA, SAPIENZA Università di Roma MOURAD OUZZANI, Qatar Computing Research Institute, Qatar Foundation ELISA BERTINO, Purdue University

With organizations increasingly depending on Web services to build complex applications, security and privacy concerns including the protection of access control policies are becoming a serious issue. Ideally, service providers would like to make sure that clients have knowledge of only portions of the access control policy relevant to their interactions to the extent to which they are entrusted by the Web service and without restricting the client's choices in terms of which operations to execute. We propose ACCONV, a novel model for access control in Web services that is suitable when interactions between the client and the Web service are conversational and long-running. The conversation-based access control model proposed in this article allows service providers to limit how much knowledge clients have about the credentials specified in their access policies. This is achieved while reducing the number of times credentials are asked from clients and minimizing the risk that clients drop out of a conversation with the Web service before reaching a final state due to the lack of necessary credentials. Clients are requested to provide credentials, and hence are entrusted with part of the Web service access control policies, only for some specific granted conversations which are decided based on: (1) a level of trust that the Web service provider has vis-à-vis the client, (2) the operation that the client is about to invoke, and (3) meaningful conversations which represent conversations that lead to a final state from the current one. We have implemented the proposed approach in a software prototype and conducted extensive experiments to show its effectiveness.

Categories and Subject Descriptors: D.4.6 [**Operating Systems**]: Security and Protection—Access control; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—Authentication; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—Web-based services

General Terms: Security

Additional Key Words and Phrases: Web services, access control, conversations

ACM Reference Format:

Paci, F., Mecella, M., Ouzzani, M., and Bertino, E. 2011. ACCONV – An access control model for conversational Web services. ACM Trans. Web 5, 3, Article 13 (July 2011), 33 pages. DOI = 10.1145/1993053.1993055 http://doi.acm.org/10.1145/1993053.1993055

© 2011 ACM 1559-1131/2011/07-ART13 \$10.00

DOI 10.1145/1993053.1993055 http://doi.acm.org/10.1145/1993053.1993055

The work of F. Paci was partly supported by the EU Commission under grant EU-FP7-FET-1P-SecureChange. The work of M. Mecella was partly supported by the EU Commission through the projects SemanticGov and SM4A11, by Regione Lazio and FILAS through the "Progetto Sensoristica Interconnessa per la Sicurezza", and by SAPIENZA Università di Roma through the grants METRO, FARI 2008, FARI 2010, and TESTMED. The work of M. Ouzzani was supported by NSF grant numbers IIS-0916614 and IIS-0811954. The work of E. Bertino was supported by the MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research.

Authors' addresses: F. Paci (corresponding author), Department of Information Engineering and Computer Science, University of Trento, Italy; email: paci@disi.unitn.it; M. Mecella, Dipartimento di Informatica e Sistemistica Antonio Ruberti, SAPIENZA Università di Roma, Italy; M. Ouzzani, Qatar Computing Research Institute, Qatar Foundation, Qatar; E. Bertino, Cyber Center, CERIAS, and Department of Computer Science, Purdue University.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from the Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

1. INTRODUCTION

Interoperation among different organizations is today a pressing need. Governmental, military, financial, medical, and private institutions increasingly participate to distributed collaborative infrastructures. Hence, distributed, heterogeneous systems are becoming very common, as globalized organizations integrate applications running on different platforms. As a consequence, there is a growing demand for architectures and technologies that support the connection and the flexible sharing of resources through the use of standardized protocols. Web services are the preferred standardsbased approach to implement distributed and heterogeneous systems. The adoption of Web services offers potential for lower integration costs and greater flexibility and interoperability.

Security is a key enabler for achieving the interoperability goal of Web service-based applications. But securing Web service-based applications is both critical and challenging. The most important security issues for Web service-based applications are related to identity management, secure message transmission, and access control. In this article we focus on the problem of how to enforce access control.

An access control model protects the Web services operations from clients that do not satisfy the authorization requirements of the Web services provider. Access control enforcement implies to find solutions to properly represent the identity of service consumers (*identification*), to verify that service consumers are who they claim to be (*authentication*), and to decide if service consumers are allowed to use Web services (*authorization*).

Since the relation between Web service consumer and Web service provider is much more loose than in traditional client-server applications, the conventional identification techniques based on a previous knowledge of users' identities such as user identifier and password are inadequate. By contrast, Web service users are usually identified by means of their own relevant properties, for example, employment status or citizenship. Such properties are typically conveyed in *digital credentials*. Digital credentials are assertions describing one of more properties about a given subject, referred to as the *owner* of the credential, certified by trusted third parties called Certification Authorities (CAs). As a consequence, *access control policies* are expressed as conditions on the properties contained in Web service users' digital credentials.

Several policy-driven access control models for Web services have been recently proposed, but none of them provides an effective access control scheme that can adequately meet the unique security challenges posed by the Web services paradigm. The most relevant proposals are by OASIS with the WS-XACML profile for Web services [2007], Wonohoesodo and Tari [2004], Sirer and Wang [2002], Bhatti et al. [2004], Emig et al. [2007], Bertino et al. [2006], Olson et al. [2006], Kagal et al. [2004], Denker et al. [2003], and Agarwal et al. [2004]. All these proposals assume that the invocation of each operation provided by a Web service is independent from the others and access control is enforced either at the level of a single Web service operation or at the level of the whole Web service.

However, in practice clients interact with Web services through a conversation process in which, at each step, a specific operation offered by the Web service is invoked. The potential operations that can be invoked depend on the state of the current conversation between the client and the Web service. As an example consider a travel agent Web service. Booking a trip at such service generally involves searching for the trip, browsing the details and rules about possible options for the trip, booking a specific package, checking out, paying, and so forth. Thus, conversations allow one to capture an important aspect of Web services, namely their "behavioral semantics."

ACM Transactions on the Web, Vol. 5, No. 3, Article 13, Publication date: July 2011.

An important requirement for a client is to be able to complete any conversation it engages with a Web service. Therefore, in order to avoid situations in which the client is not able to progress with the execution of a conversation because it does not have the proper credentials, an access control model for conversational Web services should verify upfront that a client is able to provide the credentials to be authorized to terminate the conversations the client is really interested in engaging with the Web service.

An important requirement for Web service providers is related to the disclosure of access control policies. Since access control policies can contain sensitive information, a Web service provider should not disclose access control policies until it has established sufficient trust in the client based on the credentials submitted earlier by the client [Koshutanski and Massacci 2007; Seamons et al. 2001; Yu et al. 2003]¹. Therefore, an access control model for conversational Web services should guarantee that only the access control policies of conversations that the client wants to perform are disclosed to the client.

Current approaches to access control that assume the enforcement of policies at the level of a whole Web service or of a single Web service operation do not meet the aforesaid requirements. When access control is enforced at the level of the whole Web service, a client has to provide in advance all credentials associated with all offered operations. Such an approach guarantees that a client will always be able to complete any conversation offered by the Web service. However, it has the drawback that the client will be able to know all the Web service access control policies up-front before any trust relationship has been established between the client and the Web service. Another drawback is that the client may have to submit more credentials than needed, thus violating the important security principle of the "least privilege principle" [Saltzer and Schroeder 1974]. The other approach is to require only the credentials associated with the next operation that the client wants to perform. This strategy has the advantage of requiring from the client only the credentials necessary to gain access to the requested operation. The drawback is that, after several steps, the client may reach a state from which it cannot progress because of the lack of credentials.

An alternative strategy adopted by the Semantic Web service community is based on rich service descriptions. Semantic Web services declare up front the types of credentials, defined according to some ontologies, to be possibly asked of the client [Agarwal et al. 2004]. The actual credentials are requested during the interaction between the client and the Web service. Clients can then reason in advance about compatibilities between the Web service's requirements and their own and decide if they can engage in a conversation with the Web service. While clients know only the types of the credentials to be requested, they may know enough about the access control policy of the Web service, thus making this Semantic Web service approach very similar to the first approach outlined before, that is, ask all credentials associated with all offered operations.

In this article, we present ACConv, a credential-based access control model for conversational Web services that addresses the preceding issues. ACConv has the

¹Client's credentials can contain sensitive information and they also need to be protected by access control policies. The protection of sensitive credentials from unappropriated disclosures is an interesting problem that has been widely investigated and many solutions have been proposed that can be integrated into the access control model for conversational Web services that we propose in this article. A possible solution to protect the disclosure of sensitive credentials is to establish an initial level of trust between the Web service and the client through a trust negotiation process. An alternative solution to protect sensitive credentials is to adopt Oblivious Attribute Certificates (OACerts) [Li and Li 2006], an attribute certificate scheme that allows the client to obtain a service if and only if the attribute values in the certificate satisfy the policy of the service provider, yet the service provider learns nothing about these attribute values.

following features: (1) Web service providers limit the disclosure of their access control policies to the policies that are related to the conversations the client wants to carry out with the Web service; (2) the risk for clients to be dropped out of a conversation because of the lack of necessary credentials is minimized. Note that the risk is minimized but not eliminated: clients might still not be able to progress if they want to perform an operation that is not part of the conversations for which the associated access control policies are satisfied. In line with current semantic approaches, we thus model all possible conversations as finite transition systems (a.k.a. finite state machines) [Stirling 1996; Berardi et al. 2005], in which the final states represent the states in which the interaction with the client can be ended. We refer to the conversations that lead to a final state as *meaningful conversations*. The Web service provider defines the conversation access control policy for each meaningful conversation and groups the policies in sets of policies having the same sensitivity level. The disclosure of each set of policies is protected by another type of security policy referred to as *trust policy*. When the client requests an operation, it is entrusted with the set of access control policies whose trust policy is satisfied by client's credentials and with a set of conversations referred to as *allowable conversations*. The allowable conversations are the conversations protected by the set of access control policies the client is entrusted with. The client is then requested to provide the credentials listed in the access control policies of allowable conversations starting with the execution of the operation invoked by the client. The allowable conversations starting with the execution of the operation chosen by the client and the access control policy of which is satisfied by client's credentials are referred to as granted conversations. Part of the work described in this article has been initially investigated in Mecella et al. [2006].

The article is organized as follows. Section 2 presents the main components of our conversation-based access control model for Web services. Section 3 describes the access control enforcement process for Web services, the algorithms to compute meaningful conversations, and demonstrates the correctness of our approach. Section 4 describes the architecture that implements our approach. Section 5 investigates the access control enforcement for composite Web services. Section 6 presents extensive experiments comparing our conversation-based access control enforcement, *ACConv*, with two other types of access control enforcement strategies. Section 7 discusses related work. Section 8 concludes the article. The appendix describes a complete example of a complex Web service including its access control and trust policies.

2. CONVERSATION-BASED ACCESS CONTROL

In this section, we introduce the basic concepts of Web service conversations, credentials, and access control. We then elaborate on the fundamental concepts underlying the proposed approach.

2.1 Conversational Model for Web Services

We represent the *behavioral semantics* of a Web service as the set of operations it exports and constraints on the possible conversations clients can execute. For a large class of Web services, as discussed in Benatallah et al. [2003], all such aspects can be represented as a finite transition system (refer to the WSMO choreography concept). We do not provide details about the semantic description of an operation since it is outside the scope of our work. Such semantics would be typically expressed according to a given OWL/OWL-S/WSMO ontology. Thus, a service is generally characterized by two facets: (*i*) static semantics, dealing with messages, operations, and parameters exchanged and their types, and (*ii*) dynamic semantics, dealing with the correct

sequencing of operations that represents the external workflow offered by the service. The focus of our work is on the dynamic semantics.

Definition 2.1 (Web Service)². A Web service WS is a tuple (Σ , S, s_0 , δ , F), where:

- Σ is the set of operations offered by WS; each operation in Σ denotes an interaction between WS and a client;
- -S is a finite set of states;
- $-s_0 \in S$ is the single initial state of WS;
- $-\delta \subseteq S \times \Sigma \times S$ is the service transition relation of WS;
- $-F \subseteq S$ is the set of final states of WS.

We represent $(s_i, op, s_j) \in \delta$ by $s_i \xrightarrow{op} s_j$, and we call op the *label of the transition*. The transition relation can be extended to finite length sequences of operations or conversations, defined as traces in Stirling [1996]. A *conversation* between two or more states s_1 and $s_n, s_1, s_n \in S$, is a sequence of operations $op_1 \cdot op_2 \cdot \ldots op_n, op_i \in \Sigma$, $i = 1, \ldots, n$, such that $s_1 \xrightarrow{op_1} s_2, s_2 \xrightarrow{op_2} s_3, \ldots, s_{n-1} \xrightarrow{op_n} s_n$ exist. A conversation $conv: op_1 \cdot op_2 \cdot \ldots op_n$ between two states s_1 and s_n is denoted as $s_1 \xrightarrow{op_1 \cdot op_2 \cdot \ldots op_n} s_n$.

We should observe that a Web service is often *nondeterministic* in that given a state and an action the Web service can transit to two different states. For example, in a Web service with an action pay_item, firing this action may lead to a state payment_OK, accepting the payment, or a different state payment_refused, if the credit card is not valid or the amount is above the credit limit. As a result, the client, when deciding which action to execute next, cannot be certain of which choices will be available in the next state since this depends on the transition actually executed.

2.2 Access Control Model

Credentials are the mean to establish trust between a client and a service provider. Credentials contain assertions about properties qualifying a given client, referred to as the owner. They are issued by a trusted Certification Authority (CA), which has the required domain expertise to assert that the credential owner has the set of attributes listed in the credential. The CA signs the credential with its private key so that when the credential owner uses the credential for authentication purposes, the service provider verifies the signature of the CA on the credential by using the CA's public key.

Definition 2.2 (Credential). A credential C is a tuple (Issuer, Owner, Type, Attr, Sign) where Issuer is the identifier of the CA that issues the credential, Owner is the identifier of the credential owner, Type denotes the type of the credential, Attr = { A_1, \ldots, A_n } is the set of attributes characterizing the credential type Type, and Sign is the signature of the Issuer on the whole credential. An attribute A_i is a pair (name_{A_i}, value_{A_i}), where name_{A_i} is the name of the attribute A_i and value_{A_i} is a value in the attribute domain dom_{A_i} of A_i .³

The invocation of the operations provided by a Web service is protected by *access control policies* defined by the Web service provider. These policies define conditions that clients' credentials would have to satisfy for the client to be granted the right to execute a given operation. Operation access control policies are formally defined as follows.

²From a formal point of view, we adhere to the setting proposed in Berardi et al. [2005].

³Throughout this article, we will use the dot notation to access fields of this tuple (e.g., given a credential C, *C.Issuer* represents the CA that issued the credential).

Definition 2.3 (Attribute Condition). An attribute condition *Cond* is an expression of the form: "*Type* : *name*_A op *l*", where *Type* is a credential type, *name*_A is the name of an attribute A that characterizes *Type*, op is a comparison operator such as =, <, >, \leq , \geq , \neq , and *l* is a value that can be assumed by attribute A.

Given a credential C and a condition $Cond = Type : name_{\mathcal{A}} \text{ op } l$, if (i) C.Type = Cond.Type and (ii) $\exists \ \bar{\mathcal{A}} \in C.Attr$ such that $name_{\bar{\mathcal{A}}} = Cond.name_{\mathcal{A}}$ and $value_{\bar{\mathcal{A}}}$ satisfies $Cond.(name_{\mathcal{A}} \text{ op } l)$, we say that C satisfies Cond denoted as $C \triangleright Cond$.

Definition 2.4 (*Term*). A term \mathcal{T} is a Boolean expression involving attribute conditions $Cond_1, \ldots, Cond_n$ connected by Boolean operators \vee and \wedge .

Given a set CC of credentials $\{C_1, \ldots, C_n\}$ and a term T, we say that CC satisfies T, denoted as $CC \triangleright T$, if for each $Cond_i$ in T exists $C_i \in CC$ such that C_i satisfies $Cond_i$.

Definition 2.5 (Operation Access Control Policy). Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service and *op* be an operation in Σ . An access control policy \mathcal{P} for *op* is an expression of the form "*op* if $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_n$ ", $n \ge 1$, where $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_n$ are terms.

Given a set CC of credentials $\{C_1, \ldots, C_n\}$ and an operation access control policy \mathcal{P} : op if $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$, we say that CC satisfies \mathcal{P} , denoted as $CC \vartriangleright \mathcal{P}$ if for each $\mathcal{T}_i \in \mathcal{P}, CC$ satisfies \mathcal{T}_i .

We extend the definition of operation access control policies to conversations.

Definition 2.6 (Conversation Access Control Policy). Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service, $conv : s_1 \xrightarrow{op_1 \cdot op_2 \dots op_n} s_n$ be a conversation, and $\mathcal{P}_1, \dots, \mathcal{P}_n$ be the operation access control policies for $op_1 \cdot \dots op_n$ respectively. A conversation access control policy \mathcal{P}_{conv} for conv is an expression of the form "conv if $\mathcal{P}_1, \dots, \mathcal{P}_n$ ", $n \geq 1$.

Given a set CC of credentials $\{C_1, \ldots, C_m\}$ and a conversation access control policy \mathcal{P}_{conv} : conv if $\mathcal{P}_1, \ldots, \mathcal{P}_n$, we say that CC satisfies \mathcal{P}_{conv} , denoted as $CC \triangleright \mathcal{P}_{conv}$, if for each $\mathcal{P}_i \in \mathcal{P}_{conv}, \{C_1, \ldots, C_m\}$ satisfies \mathcal{P}_i .

The definition of conversation access control policy captures the intuition that a client, owning the credentials satisfying a conversation policy, is granted access to all the operations in this conversation. If the conversation is such that it reaches a final state, then the satisfaction of the policy assures that the client will be authorized to reach the end of the conversation. The service provider will not have to deny access to some actions in the middle of the conversation because of the lack of authorization.

We assume that conversation access control policies are monotonic. This assumption is necessary to have conflict-free access control policies. We want to avoid that given a conversation access control policy conv if $\mathcal{P}_1, \ldots, \mathcal{P}_n$, a set of credentials $\{\mathcal{C}_1 \ldots \mathcal{C}_m\}$ satisfying policy \mathcal{P}_i inhibits the satisfaction of a different policy \mathcal{P}_j . Besides avoiding policy conflicts, monotonicity guarantees that policies are correct and analyzable. However, monotonicity does not allow one to express certain types of policies, most notably those that make explicit use of negative credentials such as revocation lists or other type of policies such as Chinese Wall policy and separation of duty policies. The difficulty with nonmonotonic policies is that the service provider must have the exact set of credentials from a client to make a sound access control decision. If a client knows or can predict that a certain credential will result in the decrease of its privileges, it may prefer not to reveal it. A service provider cannot distinguish whether the absence of certain credentials is caused by not having the credential or not disclosing it. To solve this problem, a possible solution is to collect credentials directly from the credential issuers rather than only from the client. However this solution introduces new problems. One problem is privacy: the issuer could disclose information about the

ACM Transactions on the Web, Vol. 5, No. 3, Article 13, Publication date: July 2011.

clients, that is, the credential, to anyone who wants the credential. It also requires issuers to be always online, which may not be practical.

With respect to credentials' validity during the execution of a conversation, we assume that once a credential has been checked (and found valid at this time), its validity is considered to last for the whole conversation it has been requested for. This means that our approach assumes *incremental consistency* as discussed in Lee and Winslett [2006]. More complex schemes, like internal, endpoint, and interval consistency, are more precise about the relationship among validity checking and effective authorization time. However, investigating them is outside the scope of the current article and left for future work.

2.3 Meaningful Conversations and Trust Policies

In our approach, clients interact with a Web service by invoking operations according to a conversation. Clients are usually interested in conversations that lead to some final states. We refer to these conversations as *meaningful conversations*.

Definition 2.7 (Meaningful Conversations). Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service and s be a state in S. The set of meaningful conversations originating from s, denoted as \mathcal{M}^s , is the set {conv | $s \stackrel{conv}{\Longrightarrow} t, t \in F$ }.

To perform meaningful conversations, clients have to provide the credentials specified by the conversation access control policies. Access control policies can contain sensitive information and should be protected from inappropriate access. Sensitive access control policies should not be disclosed until the service provider has established sufficient trust with the clients [Seamons et al. 2001; Yu et al. 2003]. Therefore, to protect conversations' access control policies, we introduce a second type of policies, called *trust policies*. Trust policies specify conditions on the credentials submitted by the client that must be satisfied to entrust the client with a set of conversation access control policies. Since conversation access control policies can have different sensitivities, the Web service provider groups them into sets based on these sensitivities. A trust policy is defined for each set of conversation access control policies.

Definition 2.8 (Trust Policy). Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service and s a state in S. Let \mathcal{M}^s be the set of meaningful conversations originating from s and Cl^s be the set of conversation access control policies associated with the conversations in \mathcal{M}^s . A trust policy that protects the access control policies in set Cl^s , denoted as $\mathcal{P}^s_{Cl^s}$, is an expression of the form " Cl^s if $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_m$ ", where $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_m$ are terms⁴.

The satisfaction of a trust policy by a client will hence define the level of trust that the Web service has on the client in state *s*. Based on the notion of trust policy, we introduce the notion of *allowable conversations*, that is, the set of meaningful conversations that are protected by the conversation access control policies which in turn are protected by the trust policy satisfied by the client. Furthermore, we introduce the concept of *granted conversations*, that are the allowable conversations that start with the execution of *op*, that is, the specific operation the client has requested.

Definition 2.9 (Allowable Conversations). Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service, s be a state in S, and CC be the set of credentials the client has submitted prior to state s. Let $\mathcal{P}_{Cl^s}^s$ be the trust policy satisfied by client's credentials in CC and Cl^s be the set of conversation access control policies protected by $\mathcal{P}_{Cl^s}^s$. The set \mathcal{A}^s of allowable

⁴Terms are defined in Definition 2.4.

ACM Transactions on the Web, Vol. 5, No. 3, Article 13, Publication date: July 2011.

conversations associated with the client in state *s* is a set of conversations defined as $\{conv \mid conv \in \mathcal{M}^s, \text{ and } \mathcal{P}_{conv} \in Cl^s\}$.

Definition 2.10 (Granted Conversations). Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service, s be a state in S, and CC be the set of credentials the client has submitted prior to state s. Let $\mathcal{P}_{Cl^s}^s$ be the trust policy satisfied by client's credentials in CC and Cl^s be the set of conversation access control policies protected by $\mathcal{P}_{Cl^s}^s$. Let \mathcal{A}^s be the set of allowable conversations associated with the client and op be the operation requested by the client in state s. The set of granted conversations associated with the client in state s and based on the operation op chosen by the client is a set of conversations defined as $\{conv \mid conv \in \mathcal{A}^s, conv = op \cdot conv', CC \triangleright \mathcal{P}_{conv}\}$. We denote this set as $\mathcal{G}^s \mid_{op}$.

To summarize, in our approach we consider three different sets of conversations.

- The set of meaningful conversations associated with a state s of WS is static and computed offline (not at enforcement time). Its computation is necessary for the Web service provider to determine the conversations for which it has to define access control policies. Once access control policies are defined, the Web service provider groups the policies according to their sensitivity.
- The set of allowable conversations is dynamically associated with a client in a given state of the interaction with the Web service. The Web service provider assigns the client a set of conversation access control policies, the trust policies of these are satisfied by the client's credentials. The meaningful conversations protected by the conversations's access control policies the client is entrusted with are the allowable conversations.
- The set of granted conversations is dynamically associated with a client in a given state of the interaction with the Web service. The set of granted conversations is the subset of allowable conversations starting with the operation chosen by the client, and for which the client's credentials satisfy the corresponding conversation access control policies.

3. ACCESS CONTROL ENFORCEMENT

In this section, we describe two important aspects of *ACConv*: computation of meaningful conversations and *ACConv*'s access control enforcement protocol.

3.1 Meaningful Conversations Computation

To enforce the proposed access control model, a Web service provider has to compute for each state s of the Web service all the possible meaningful conversations \mathcal{M}^s . Once, for a state s, the meaningful conversations \mathcal{M}^s are determined, the Web service provider specifies the access control policies for each operation and then derives the conversation access control policies can have different sensitivity according to the level of protection required by the conversations. Therefore, the Web service provider groups the conversation access control policies associated with state s in sets Cl^s of policies having the same sensitivity level. The disclosure of each set of access control policies Cl^s is protected by a trust policy $P_{Cl^s}^s$ that is defined by the Web service provider. The number of meaningful conversations for a state s is finite if the Web service

The number of meaningful conversations for a state s is finite if the Web service transition system is acyclic. The meaningful conversations can be computed by a simple breadth-first traversal of the transition system. However, if the transition system contains cycles, the number of meaningful conversations can be potentially infinite and so is the number of conversation access control policies. Fortunately, if a client performs a conversation that involves a cycle, the service provider has to verify that

ACM Transactions on the Web, Vol. 5, No. 3, Article 13, Publication date: July 2011.

the client is authorized to invoke the operations in the cycle only one time. Since we assume that once client's credentials have been checked, they are valid for the whole conversation execution they have been requested for. Thus, the meaningful conversations in which a cycle is traversed only once are equivalent to the conversations in which a cycle is traversed an infinite number of times.

Based on the preceding assumption, we propose an algorithm to compute all the possible meaningful conversations for each state of a Web service transition system. The algorithm is based on the concept of *strongly connected component* (SCC for short). A strongly connected component is the maximal subgraph of a directed graph such that for every pair of vertexes (u, v) in the subgraph, there is a directed path from u to v and a directed path from v to u [Tarjan 1972]. The transition system of a Web service can be considered as a directed graph where a transition between two states is a directed edge without the label. Therefore, a new acyclic graph can be generated whose nodes represent the different strongly connected components of the initial Web service transition system WS. In the new graph, the cycles are "collapsed" into strongly connected components while the states which are not involved in cycles will remain unchanged in the new graph. In what follows, we denote as c(s), the strongly connected component to which a state *s* belongs to.

The graph of the strongly connected components can be formally defined as follows.

Definition 3.1 (Directed Graph of the Strongly Connected Components). Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service. The directed graph of the strongly connected components for WS is a graph $\mathcal{G}^{SCC} = \langle N^{SCC}, E^{SCC} \rangle$ where $N^{SCC} = \{c \mid c \text{ is a strongly connected component in } WS\}$ is the set of vertexes, and $E^{SCC} = \{\langle c_1, c_2 \rangle \mid c_1, c_2 \in N^{SCC}, c_1 \neq c_2, \exists s_1 \xrightarrow{op} s_2 \in \delta, op \in \Sigma, s_1, s_2 \in S, c_1 = c(s_1), c_2 = c(s_2)\}$ is the set of direct edges.

 \mathcal{G}^{SCC} can be efficiently computed through the classical Tarjan's algorithm [Tarjan 1972] or more recent optimizations [Nuutila and Soisalon-Soininen 1993].

To compute the meaningful conversations we exploit the notion of directed graph of strongly connected components and the following properties of strongly connected components.

Definition 3.2 (Ingoing and Outgoing Nodes). Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service. Let $\mathcal{G}^{SCC} = \langle N^{SCC}, E^{SCC} \rangle$ be the graph of strongly connected components of WS and *scc* be a strongly connected component in N^{SCC} . A state $s \in S$ such that c(s) = scc is an *in-going* node of *scc* if $\exists s_1 \xrightarrow{op} s \in \delta$, $op \in \Sigma$, $s_1 \in S$, $c(s_1) \neq scc$. A state $s \in S$ such that c(s) = scc is an *out-going* node of *scc* if $\exists s \xrightarrow{op} s_1$, $op \in \Sigma$, $s_1 \in S$, $c(s_1) \neq scc$.

 $\begin{array}{l} Definition \ 3.3 \ (Cardinality). \ \mbox{Let} \ \mathcal{WS} = (\Sigma, S, s_0, \delta, F) \ \mbox{be a Web service. \ Let} \ \mathcal{G}^{SCC} \\ = \langle N^{SCC}, \ E^{SCC} \rangle \ \mbox{be the graph of strongly connected components for} \ \mathcal{WS} \ \mbox{and} \ scc \ \mbox{be a strongly connected component in} \ N^{SCC}. \ \mbox{The cardinality of scc, denoted as } card(scc), \ \mbox{is the cardinality of the set} \ \mathcal{O}_{scc} = \{op \in \Sigma \mid \exists s_1 \xrightarrow{op} s_2 \in \delta, \ s_1, s_2 \in S, \ c(s_1) = c(s_2) = scc \ \}. \end{array}$

Definition 3.4 (Covering Traversing Path). Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service. Let $\mathcal{G}^{SCC} = \langle N^{SCC}, E^{SCC} \rangle$ be the graph of strongly connected components for WS and scc be a strongly connected component such that $scc \in N^{SCC}$. Let I_{scc} be the set of ingoing nodes of scc, O_{scc} be the set of scc outgoing nodes, N^{scc} be the set of nodes of scc, and E^{scc} be the set of arcs of scc. A path s_1, \ldots, s_n is a covering traversing path for scc if it satisfies the following conditions: (a) $s_1 = u, s_n = v, (u, v) \in I_{scc} \times O_{scc}$, (b) is the shortest path between nodes u and v, (c) $\forall s_i, \exists s_i \xrightarrow{op} s_{i+1} \in \delta, op \in \Sigma, s_i, s_{i+1} \in S, c(s_i) = c(s_{i+1}) = scc$, (d) $\forall e: s_k \xrightarrow{op} s_j \in E^{scc}, \exists s_i \xrightarrow{op} s_{i+1}, s_i, s_{i+1} \in N^{scc}, i = 1 \ldots n, op \in \mathcal{O}_{scc}$.

ALGORITHM 1: compute()

Input: s: State; WS: Web Service; conv: Conversation; Meaningful_Conversation: SetOfConversation

Output: Meaningful_Conversation: SetOfConversation

(1)	if <i>s</i> has no out-going transition (i.e., is a leaf)
(2)	if isNewConversation(conv)
(3)	$Meaningful_Conversation.add(conv);$
(4)	return (Meaning ful_Conversation);
(5)	else
(6)	$\mathbf{if} \mathit{conv}.\texttt{length}() > \texttt{rank}(\mathit{c(s)})$
(7)	return (<i>Meaningful_Conversation</i>);
(8)	else
(9)	if $s \in WS.F$ (i.e., s is final) AND isNewConversation(<i>conv</i>)
(10)	Meaningful_Conversation.add(conv);
(11)	foreach $s \xrightarrow{a} t$
(12)	$compute(t, conv \cdot a, Meaningful_Conversation);$

Definition 3.5 (Coverage). Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service. Let $\mathcal{G}^{SCC} = \langle N^{SCC}, E^{SCC} \rangle$ be the graph of strongly connected components for WS and scc be a strongly connected component such that $scc \in N^{SCC}$. Let I_{scc} be the set of ingoing nodes of scc and O_{scc} be the set of scc outgoing nodes. Let CTS_{scc} be the set $\{s_1, \ldots, s_n \mid s_1, \ldots, s_n \text{ is a covering traversing path for <math>scc\}$. The coverage of scc is the min $\{n \in \mathbb{N} \mid n = length(s_1, \ldots, s_n), s_1, \ldots, s_n \in CTS_{scc}\}$.

Definition 3.6 (Rank). Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service. Let $\mathcal{G}^{SCC} = \langle N^{SCC}, E^{SCC} \rangle$ be the graph of strongly connected components for WS and *scc* be a strongly connected component such that $scc \in N^{SCC}$. The rank of *scc*, denoted as *rank(scc)* is defined as follows.

 $rank(scc) = \begin{cases} coverage(scc) & \text{if } scc \text{ is the root of } \mathcal{G}^{SCC} \\ 1 + coverage(scc) + max(rank(m)) & \text{where } m \text{ is the number of all the possible} \\ predecessors of scc \end{cases}$

Details on how the coverage and rank of a strongly connected component can be computed can be found in Gonnet and Baeza-Yates [1991].

Intuitively, the rank expresses the maximum length of the meaningful conversations in which a cycle is traversed only once.

The compute() algorithm computes for the given state s the set of meaningful conversations \mathcal{M}^s . The algorithm uses the concept of rank to avoid possible infinite recursion. The isNewConversation() algorithm checks if a given conversation is already in the *Meaningful_Conversation* set, that is, the set of meaningful conversations for the given state s.

3.2 Enforcement Protocol

Our access control enforcement protocol ensures that Web service providers disclose to clients only the access control policies of conversations these clients want to perform.



Fig. 1. Access control enforcement process.

Therefore, clients have to provide only the credentials necessary to be granted the execution of the conversations they are interested in.

In fact, the Web service providers ask clients to provide the credentials specified in the access control policies of allowable conversations starting with the operation chosen by the clients. If clients provide the requested credentials, they can perform any of the allowable conversations starting with the chosen operation. Since clients are requested to provide in advance all the credentials in order to be authorized to perform conversations starting with the operation they have selected, the risk that clients are not able to progress the interaction with the Web service is minimized. The risk is minimized but not eliminated: clients might still not be able to progress if they want to perform an operation that is not part of the granted conversations. In this case, our enforcement protocol tries to entrust clients with another set of access control policies and with another set of granted conversations.

The enforcement process is represented in Figure 1. The enforcement is triggered whenever a Web service receives an invocation of an operation Op from a client. If Op is included in the set of granted conversations, Op is executed and the result is returned to the client. Otherwise, the credentials presented by the client are evaluated against the trust policies of the set of access control policies protecting conversations starting with Op. If the client's credentials satisfy a trust policy, the client is associated with a set of access control policies and with the corresponding set of allowable conversations. Thus, the client is requested to provide the credentials specified in the access control policies of the subset of allowable conversations starting with Op. If the client is requested to provide the credentials specified in the access control policies of the subset of allowable conversations starting with Op. If the client is granted the execution of the allowable conversations starting with Op, that are the granted conversations. If the client does not match any trust policy, the client is entrusted only with the access control policy of operation Op. Therefore, the client is asked to provide only the credentials required by

Op access control policy. When the client requests the next operation, the enforcement system tries to entrust the client with a set of access control policies and of granted conversations on the basis of the current operation requested.

The Enforcement() algorithm represents the overall enforcement process. The first step is to check if the current operation Op is contained in the set of operations composing the set *Granted_Conversations*. If this is the case, the operation is executed. Otherwise, TrustAssignment() is executed to check if the client can be entrusted with a set of conversation access control policies and of allowable conversations. If this is not the case, the client is entrusted only with Pol_{Op} , the operation access control policy associated with Op. Then, the method Select_Cred returns the set of credentials listed in the policy Pol_{Op} that the client has not yet provided. Thus, the client is requested to submit only the credentials in the set returned by Select_Cred. If the client's credentials in set $Cred_Set$ satisfy Op policy, Op is performed; otherwise the client is denied the execution of Op.

If the client is entrusted with a set of conversation access control policies Pol_Set and a set of allowable conversations $Conv_Set$, the algorithm first computes the subset Conv of $Conv_Set$ containing the conversations that start with the execution of Opand the subset Pol of Pol_Set containing the access control policies of conversations in Conv. Then, the client is asked to provide the credentials listed in the access control policies in Pol. The set Conv of conversations which access control policies are satisfied by the client's credentials becomes the new set of granted conversations. If the client does not satisfy any of the policies in Pol, the algorithm checks if client's credentials satisfy the access control policy associated with Op. If this is the case, the execution of Op is granted to the client, otherwise the interaction with the client is terminated. The algorithm isGranted() checks if the operation Op invoked by the client is contained in the set of operations composing the conversations in set $Granted_Conversations$.

The TrustAssignment() algorithm determines which trust policy $P_{Cl_i^s}^s$ is satisfied by *Client_Cred*, that is, the set of credentials submitted by the client prior to state s. If the client's credentials satisfy a trust policy $P_{Cl_i^s}^s$, the client is entrusted with the set of access control policies Cl_i^s and with the set of allowable conversations $A_{Cl_i^s}^s$. Otherwise, the client is entrusted only with the access control policy P_{Op} associated with the operation Op chosen by the client and with an empty set of allowable conversations.

We refer the reader to Appendix A for a complete example of enforcement based on operation and conversation access control policies defined for the Amazon Flexible Payment Web service.⁵

3.3 Validation

The following theorems prove the correctness of our approach by showing that a client can execute only the meaningful conversations for which the client satisfies the conversation access control policies (Theorem 3.7) and a client cannot be denied the execution of a conversation the client should be granted access to (Theorem 3.8).

THEOREM 3.7. Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service, s be a state in S, and CC be the set of credentials the client has already submitted. Let $\mathcal{P}_{Cl^s}^s$ be the trust policy satisfied by client's credentials in set CC and Cl^s be the set of conversation access control policies protected by $\mathcal{P}_{Cl^s}^s$. Let \mathcal{A}^s be the set of allowable conversations associated with the client in state s and Op be the operation requested by the client in state s. Let $\mathcal{G}^s \mid_{Op} = \{conv \mid conv \in \mathcal{A}^s, conv = Op \cdot conv', CC \triangleright \mathcal{P}_{conv}, \mathcal{P}_{conv} \in Cl^s\}$ be

⁵Refer to http://aws.amazon.com/fps.

ACM Transactions on the Web, Vol. 5, No. 3, Article 13, Publication date: July 2011.

the set of granted conversations. Let SatisfiedPol = { $\mathcal{P}_{conv} \mid \mathcal{P}_{conv} \in Cl^s$, $conv \in \mathcal{G}^s \mid_{Op}$, $\mathcal{CC} \triangleright \mathcal{P}_{conv}$ } be the set of conversation access control policies that are satisfied by the client's credentials in set \mathcal{CC} and that protect the granted conversations. The client can perform only the meaningful conversations in $\mathcal{G}^s \mid_{Op}$.

PROOF. The proof of this theorem is by contradiction. Suppose that the client is granted the execution of a conversation $conv'' = Op \cdot conv'$ that is not in the set of granted conversations $\mathcal{G}^s |_{Op}$. Since the client is authorized to perform conv'', there exists a conversation access control policy $\mathcal{P}_{conv''}$ such that the set of client credentials \mathcal{CC} satisfies $\mathcal{P}_{conv''}$, denoted as $\mathcal{CC} \triangleright \mathcal{P}_{conv''}$. Therefore, $\mathcal{P}_{conv''}$ must belong to the set SatisfiedPol and conv'' must belong to $\mathcal{G}^s |_{Op}$. This contradicts our assumption and thus the demonstration of the theorem follows.

THEOREM 3.8. Let $WS = (\Sigma, S, s_0, \delta, F)$ be a Web service, s be a state in S, and CC be the set of credentials the client has already submitted. Let $\mathcal{P}_{Cl^s}^s$ be the trust policy satisfied by client's credentials in set CC and Cl^s be the set of conversation access control policies protected by $\mathcal{P}_{Cl^s}^s$. Let \mathcal{A}^s be the set of allowable conversations associated with the client in state s and Op be the operation requested by the client in state s. Let $\mathcal{G}^s \mid_{Op} = \{conv \mid conv \in \mathcal{A}^s, conv = Op \cdot conv', CC \triangleright \mathcal{P}_{conv}, \mathcal{P}_{conv} \in Cl^s\}$ be the set of granted conversations. Let SatisfiedPol = $\{\mathcal{P}_{conv} \mid \mathcal{P}_{conv} \in Cl^s, conv \in \mathcal{G}^s \mid_{Op}, CC \triangleright \mathcal{P}_{conv}\}$ be the set of conversation access control policies that are satisfied by the client's credentials in set CC and that protect the granted conversations. The client cannot be denied the execution of granted conversations in $\mathcal{G}^s \mid_{Op}$.

PROOF. The proof of this theorem is by contradiction. Assume that the client wants to perform a granted conversation $conv'' = Op \cdot conv'$ but the execution of this conversation is denied to the client. The execution of conv'' is denied to the client only if the client's credentials in the set CC do not satisfy $\mathcal{P}_{conv''}$, that is, the conversation access control policy of conv''. But, because conv'' is one of the granted conversations, $\mathcal{P}_{conv''}$ must belong to *SatisfiedPol* set, that is, the set of conversation access control policies that are satisfied by client's credentials in set CC. This contradicts the initial assumption and thus the demonstration of the theorem follows.

4. IMPLEMENTATION OF THE ENFORCEMENT SYSTEM

4.1 Architecture

The system architecture of *ACConv* is compliant with the XACML standard [Moses 2005]. The main components of the *access control enforcement system* are a Policy Enforcement Point (PEP), a Policy Decision Point (PDP), and a Policy Administration Point (PAP). With respect to the XACML architecture, we have added a component called Execution Controller System (ECS), see Figure 2.

The ECS provides the security administrator with a simple graphical interface, through which she can compute the meaningful conversations and define trust and access control policies. In particular, the ECS interface allows the security administrator to specify the access control policies for the operations provided by the Web service. Then, the ECS combines operation access control policies to obtain the conversation access control policies. Once the conversation access control policies are computed, the security administrator can group the policies based on their sensitivity and can define a trust policy to protect the release of each set of policies. Access control policies and trust policies are then stored in two repositories that are managed by the PAP component. Besides providing functions to set up the enforcement access control process, the ECS also tracks, at runtime, the state of the interaction between the client and the service.

ALGORITHM 2: Enforcement()

Input: s: State

Op: Operation /* Operation chosen by the client */ Granted_Conversations: SetOfConversation /* Set of granted conversations */ $\begin{array}{l} AC_Pol: \texttt{SetOfSetofACPol}/\texttt{Set} \{Cl_1^s \ldots Cl_n^s\} \text{ of sets of access control policies associated with state $s^*/$ $TP_Set: \texttt{SetOfTrustPol}/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trust policies associated with state $s^*/$ $TP_Set: SetOfTrustPol/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trust policies associated with state $s^*/$ $TP_Set: SetOfTrustPol/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trust policies associated with state $s^*/$ $TP_SetOfTrustPol/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trust policies associated with state $s^*/$ $TP_SetOfTrustPol/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trust policies associated with state $s^*/$ $TP_SetOfTrustPol/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trustPolicies associated with state $s^*/$ $TP_SetOfTrustPol/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trustPolicies associated with state $s^*/$ $TP_SetOfTrustPol/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trustPolicies associated with $tate $s^*/$ $TP_SetOfTrustPol/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trustPolicies associated with $tate $s^*/$ $TP_SetOfTrustPol/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trustPolicies associated with $tate $s^*/$ $TP_SetOfTrustPol/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trustPolicies associated with $tate $s^*/$ $TP_SetOfTrustPol/\texttt{Set} of $\{P_{Cl_1^s}^s \ldots P_{Cn_n^s}^s\}$ trustPolicies associated with $tate $s^*/$ $TP_SetOfTrustPol/SetOfTrustPo$ Client_Cred: SetOfCredential /* Credentials already submitted by the client */ Output: Result: Message (1) var Conv_Set: SetOfConversation; (2)var Cred_Set: SetOfCredentialType (3)var Pol_Set: SetofACPol; (4)var Op_Set: SetofOperation; (5)var Granted: Boolean; (6)Receive(Execute(OP)); (7)Granted := isGranted(s, Op, Granted_Conversations); (8) if Granted (9)Result := Execute(Op); (10)Return(*Result*); (11)else (12){Conv_Set, Pol_Set} := TrustAssignment(s, Op, AC_Pol, TP_Set, Client_Cred); (13) $if Conv_Set == \emptyset$ (14) $Pol := \{Pol_{Op}\};$ (15) $\{C_1, \ldots, C_i\} := \text{Select_Cred}(Pol_{Op}, Client_Cred);$ (16)Request_Cred({ $\mathcal{C}_1, \ldots, \mathcal{C}_j$ }); (17)Sub mitted_Cred :=Receive_Cred; Client_Cred.add(Sub mitted_Cred); (18)if Client_Cred ▷ Pol_{Op}
 Result := Execute(Op); (19) (20) (21)Return(Result); (22)else (23)Result := "Fault"; (24)Return(Result); (25)else (26) $Conv := \{ conv \mid conv \in Conv \, \exists et \wedge conv = Op \cdot conv' \};$ $Pol := \{ Pol_i \in Pol_Set | Pol_i = Pol_{conv} \land conv \in Conv \};$ (27)foreach $Pol_i \in Pol$ (28)(29) $\{C_1, \ldots, C_j\} := \text{Select_Cred}(Pol_i, Client_Cred);$ $Cred_Set.add(\{\mathcal{C}_1,\ldots,\mathcal{C}_j\});$ (30) (31) Request_Cred(Cred_Set); Sub mitted_Cred :=Receive_Cred; (32)(33)Client_Cred.add(Sub mitted_Cred); (34) **foreach** $Pol_i \in Pol$ **if** Client_Cred \triangleright Pol_i (35)Granted_Conversations.add({ $conv \in Conv | Pol_i = Pol_{conv}$ }); (36)(37)**if** Granted_Conversations $\neq \emptyset$ (38)Result := Execute(Op):(39) Return(Result); (40)else if $Client_Cred \triangleright Pol_{Op}$ (41) Result := Execute(Op); (42)Return(*Result*); (43)(44)else Result := "Fault"; (45)(46)Return(Result);

The PEP Module is the interface between the Web service's clients and the ECS. According to the enforcement process described in Section 3, when the client invokes the first operation, it sends a request message along with a set of initial credentials. Once the PEP receives the message, it stores the credentials in a local repository and forwards to the ECS the name of the operation selected by the client. The ECS updates the state of the interaction and returns to the PEP the identifiers of the possible sets of access control policies the client can be entrusted with in that state and the corresponding sets of allowable conversations (steps 2–3). Then, the PEP sends to the

ACM Transactions on the Web, Vol. 5, No. 3, Article 13, Publication date: July 2011.

ALGORITHM 3: isGranted()

Input:

s: State Op: Operation/* Operation chosen by the client */ Granted_Conversations: SetOfConversation /* Set of granted conversations */ **Output:** Granted: Boolean;

1)	
	var <i>Granted</i> : Boolean;
2)	Granted := false;
3)	foreach $Conv \in Granted_Conversations$
4)	OpSet := getOperation(Conv);
5)	$Granted_Operations.add(OpSet);$
6)	if $Op \in Granted_Operations$
7)	Granted := true;
8)	else
9)	Granted := false;
10)	return(<i>Granted</i>);

PDP the identifier of the sets of conversation access control policies and the sets of allowable conversations (step 4).

The PDP's Trust Level Assignment (TLA) module interacts with the PAP which manages the policies, to retrieve the trust policies associated with the current state. Then, it queries the credential repository to evaluate clients' credentials against trust policies. The client is entrusted with the set of conversation access control policies of which the trust policy is satisfied. The TLA module notifies to the Policy Selection (PS) module the identifier of the set of policies the client has been associated with and the set of allowable conversations. The PS module asks the PAP to retrieve the conversation access control policies in the policies set assigned to the client (steps 8-9). Then, the PS module returns the set of policies to the PEP with the set of allowable conversations (step 10). Thus, the PEP asks the client to provide the credentials required by the policies that it has not yet provided and evaluates them against the policies (steps 11-12). If the client's credentials satisfy the policies, the client can perform any operation that composes the granted conversations. Then, the PEP triggers the execution of the operation by the internal middleware and returns the result to the client (steps 13-15). As the PEP stores the granted conversations, every time the client invokes an operation, it first checks if the operation is one of those that compose the granted conversations. If this is the case, the PEP sends to the ECS the name of the requested operation and then triggers the execution of the operation. Otherwise, the PEP requests the ECS to provide the set of the identifiers of the sets of access control policies, and the sets of allowable conversations the client can be entrusted with in the current state of the interaction.

4.2 Implementation Details

The preceding architecture has been implemented using Globus Toolkit 4 [Globus 2011]. To achieve secure communications between the client and the server hosting the Web service, the system uses the GRID Security Infrastructure (GSI) secure message

ALGORITHM 4: TrustAssignment()

Input:

s: State

Op: Operation /* Operation chosen by the client */

 $\{Cl_1^s\dots Cl_n^s\}$: SetOfSetofACPol /*Set of sets of access control policies associated with state s */

 $\{P_{Cl_1^s}^s \dots P_{Cl_n^s}^s\}$: SetOfTrustPol/*Set of trust policies associated with state $s * / Client_Cred$: SetOfCredential /* Credentials already submitted by the client */

Output:

Conv_Set: SetOfConv /*Set of allowable conversations*/ Pol_Set: SetofACPol /*Set of conversation access control policies assigned to the client*/

(1)var Conv_Set: SetOfConv; (2)var Pol_Set: SetofACPol var Satisfied_Pol: TrustPol; (3)(4) $Conv_Set := \emptyset;$ $\begin{array}{l} \textbf{foreach} \ P^{s}_{Cl^{s}_{1}} \in \{P^{s}_{Cl^{s}_{1}} \dots P^{s}_{Cl^{s}_{n}}\} \\ \textbf{if} \ Client_Cred \rhd P^{s}_{Cl^{s}_{i}} \end{array}$ (5)(6)(7) $Pol_{-}Set.add(Cl_{i}^{s});$ $Conv_Set.add(A^s_{Cl^s});$ (8)Satisfied_Pol := $P_{Cl^s}^s$; (9) (10)break; if Satisfied_Pol == null (11)(12) $Pol_Set.add(\{P_{Op}\});$ (13)**return** {*Conv_Set*, *Pol_Set*};

mechanism, which is based on the WS-Security specification [Lawrance and Kaler 2006].

The system uses the XACML standard version 1.1 for representing the operation access control policies and trust policies. This version is implemented by Sun's XACML implementation [Sun 2003]. Moreover, the system uses the SAML standard version 1.1 for representing the credentials. This standard is implemented from OpenSAML [Internet2 2006]. A Web service transition system is represented by an XML file, according to a specific XML Schema.

The software implementing our system is organized into four components: (i) the PDP module which is developed as a custom GSI authorization mechanism; the TLA and PS modules used by the PDP, which are implemented as Java classes; (ii) the PAP module which is a Java class used by the TLA for retrieving the trust policies and by the PS for retrieving the conversation access control policies; (iii) the service module which contains a Web service used by the client to communicate with the system; and also includes a Java superclass that a given Web service class must extend for being able to use the system; (iv) the handler module, which contains a JAX-RPC handler that allows one to extract from the WS-Security header, contained in the incoming messages, the information required by the system (in particular the SAML assertions).

On the client side, to use a Web service deploying our system, two software components are needed: (i) a client module which contains the client-side logics for carrying



Fig. 2. System architecture for a single Web service.

out the access control protocol (as described in Section 3), and (ii) a handler module which inserts into outgoing messages the information required from the system (in particular the WS-Security header and the SAML assertions).

With respect to a traditional Web service platform, the implemented system is a kind of "wrapper" of the actual Web service. It intercepts every request to the Web service and requires a description of the transition system associated with the Web service and a set of access policies (operation access control policy and trust policies). The description of the transition system can be specified by the Web service developer, while the access policies can be specified by the security administrator. With respect to the client side, we only require the initialization of the component that manages the requests for credentials. This component uses the credentials associated with the client, which can be issued by a SAML Authority or can be generated by the user.

To prepare a Web service to use our system, the steps to follow are: (i) the service developer defines the service's interface using WSDL; (ii) the service developer writes the XML document describing the transition system of the Web service; (iii) the security administrator defines the access policies for the Web service; (iv) the service developer implements the service (using Java in our case); (v) the service developer generates a GAR file (Globus Archive) and deploys it with a GT4 tool (this step depends on the Globus Toolkit); (vi) the service client uses the WSDL file to generate proxy client classes; and (vii) the service client initializes the component that manages the requests and gives that component access to client's credentials. The client can now communicate with the Web service protected with the proposed system.

5. ACCESS CONTROL FOR COMPOSITE WEB SERVICES

A key strength of Web services is at they can be composed to build new Web services called *composite services*. The Web services being composed are usually referred to as *component services*. Composition involves two different aspects [Berardi et al. 2005]:

13:18

(1) synthesis which is concerned with producing a specification, called *composition* schema, of how to coordinate the component services to fulfill the client request; and (2) orchestration which relates to the enactment of the composite service and the coordination among component services by executing the composition schema. The composition schema can be generated manually by a designer or (semi-)automatically [Berardi et al. 2005]. An orchestration engine can then invoke the component services according to the schema. In many cases, the behavior of the composite service and the component services can be modeled using transition systems. A transition system is also used to represent the composition schema [Berardi et al. 2005].

5.1 Enforcement

The composite service does not take any authorization decision during the enforcement process. Instead, the component services which provide the operations invoked by the client decide whether the execution has to be granted. Each component service determines the set of conversation access control policies the client can be entrusted with based on the trust policy it satisfies. Once the client is assigned to a set of conversation access control policies, the enforcement process is the same as the one for single Web services described in Section 3. However, there are few differences. First, the client does not directly communicate with the component services but with the composite service. The composite Web service has to forward the client's requests to the component services and vice versa and caches the credentials a client submits to the component services. The caching mechanism avoid to ask credentials that the client has already provided and that are not expired. The intermediary role played by the composite Web service requires that the composite Web service is trusted both by the client and the component services. In fact, the client hands over its credentials to the composite Web service and delegates to it the decision to whom these credentials can be disclosed. The trust relationship between the client and the composite Web service can be created by carrying out a trust negotiation when the client starts the interaction with the composite Web service.

It is important that there is a trust relationship also between the composite Web service and the component Web services because, on one side, the composite Web service needs to be sure that it is not disclosing client credentials to a malicious Web service, and, on the other side, the component Web services need to establish sufficient level of trust in the composite Web service before partially revealing their access control policies to the composite service. To bootstrap such a relationship, the composite Web service and the component service carry out a trust negotiation when the composite Web service forwards to the component Web service the request of the client to invoke an operation that the component Web service provides. Another difference between the enforcement process for single and composite Web services is that the client is assigned to a different set of conversation access control policies by each component service rather than just one. Moreover, the set of allowable conversations assigned by a component Web service to the client might include conversations that are not part of the composition schema. If this is the case, the client is asked to provide only the credentials for the allowable conversations that are composed of operations that are in the composition schema. This avoids that the client has to provide credentials for conversations she will never invoke because they are not provided by the composite Web service.

5.2 Architecture

The system architecture for composite Web services is composed of multiple access control enforcement systems, one for each component service, and of a repository for



Fig. 3. System architecture for a composite Web service.

storing clients' credentials (see Figure 3). The structure of the access control system for the component Web services is similar to the system that we have described in Section 4.1 for single Web services.

The orchestration engine has two functions: it manages clients' credentials and invokes the component Web services's operations necessary to fulfill a clients' request. When a client invokes an operation (step 1), the orchestration engine first stores in its local repository the credentials the client sent with the invocation. Then, the orchestration engine contacts the component service enforcement system that according to the composition schema is entitled to perform the operation. The PEP of the component service checks if the client is authorized to execute the operation. If this is the case, the operation is performed; otherwise the PDP returns to the PEP the entrusted conversations' access control policies. The PEP sends to the orchestration engine the request for the credentials specified in the conversation access control policies (step 3). The orchestration engine, then, determines which of the credentials, among those requested by the component Web services, are stored in its local credential repository and thus only requests to the client the credentials that the client has not yet provided (step 4). Once the client provides the credentials (step 5), the orchestration engine sends them to the component service's PEP (step 6). If the client's credentials are compliant with the access control policies, the operation is executed and the result is returned to the client through the orchestration engine (steps 7–8). Otherwise, the access is denied.

6. VALIDATION AND EXPERIMENTS

In this section, we describe the experiments we have run to evaluate whether the *ACConv* enforcement strategy strikes a good balance between the single-operation strategy and the request-all strategy.

On one hand, the request-all strategy requires the client to provide in advance all the necessary credentials in order to be authorized to perform all the conversations supported by a Web service. With this approach, the client has to submit credentials also related to conversations that it will never perform, but it is sure it will be able to complete any conversation it carries on with the Web service.

On the other hand, the single-operation strategy requires the client to provide only the credentials needed to satisfy the access control policy of the operation the client invokes during the interaction with the Web service. Therefore, the client is not requested to provide unnecessary credentials, but it might not be able to complete the execution of a conversation because it does not have the necessary credentials. In contrast, our approach should strike a good balance between asking to clients the credentials needed to be authorized to progress the interaction with the Web service, and giving assurance to clients that they can eventually reach a final state.

6.1 Relevant Parameters

We evaluate ACConv according to three parameters: the loss, the number of credential disclosures, and the number of credential requests. The loss is the number of operations that have been executed by a client before being dropped out of a conversation because it does not have the credentials necessary to perform the subsequent operations. Executing an operation requires the service provider to allocate resources, and if the conversation is suddenly interrupted in a nonfinal state, all these resources have been "wasted." The number of credential disclosures measures the number of credentials clients have to provide in order to be authorized. The number of credential requests indicates the number of times clients are solicited to provide credentials.

To evaluate whether the ACConv enforcement process is a good trade-off between request-all and single-operation strategies, we also compute the loss ratio, the number of credential disclosures ratio, and the number of credential requests ratio. The loss ratio is the ratio $\frac{loss_{ACCOnv}}{loss_{single-op}}$ where $loss_{ACCOnv}$ and $loss_{single-op}$ are the loss of the AC-Conv strategy and the single-operation strategy respectively. The number of credential disclosures ratio is the ratio $\frac{NumCred_{ACCOnv}}{NumCred_{request-all}}$ where $NumCred_{ACCOnv}$ is the number of credential dential disclosures of the ACConv and NumCred_{request-all} is the number of credential disclosures for request-all strategy. We compute the loss and the number of credential disclosures ratios in this way because the maximum possible loss is generated by the single-operation strategy whereas the request-all strategy maximizes the number of credential disclosures. To compare the number of credential requests in our strategy with the number of requests of request-all and single-operation strategies, we compute the number of credential requests ratio as the ratio $\frac{NumCredReq_{ACConv}}{NumCredReq_{request-all}}$ ($\frac{NumCredReq_{ACConv}}{NumCredReq_{single-op}}$) between ACConv strategy's number of credential requests and request-all (singleoperation) number of credential requests.

We now show how the loss, the number of credential disclosures, and number of credential requests can be computed for each access control enforcement strategy. We make the following assumptions:

- -WS is a Web service providing *n* operations; $-conv: s_1 \xrightarrow{op_1 \cdot op_2 \dots op_n} s_n$ is a conversation of *n* different operations $op_1, op_2, \dots op_n$ provided by WS;
- -j is the number of operations that compose the granted conversations;
- each operation access control policy op_i if $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_l, i = 1, \ldots, n$ is such that each term \mathcal{T}_k , k = 1, ..., l is an attribute condition $Cond = \mathcal{T}ype : name_{\mathcal{A}} \text{ op } v$ and $\mathcal{T}ype$ is different for each *k*.

Request-All. As the client has provided the credentials to be authorized to perform all the operations supported by the Web service, the loss associated with conversation *conv* by the request-all strategy is equal to 0. Since each access control policy states conditions on l different credential types, the number of credential disclosures associated with conv is l^*n . The number of credential requests is equal to 1 since the client is

requested to provide all the necessary credentials to be authorized and provides them at once.

Single-Operation. The loss associated with conv, when the single-operation strategy is adopted, is equal to $loss_{op_i} = (i-1)$ where op_i is the operation of conv that the client is not authorized to perform. In other words, the loss is represented by the execution of the operations that precede op_i in the conversation, that is op_1, \ldots, op_{i-1} . The number of credential disclosures associated with conv is l^*n . Since the client is requested to provide the credentials each time it requests an operation, the number of credential requests in this case is equal to n.

ACConv. If conv is one of the granted conversations assigned to the client, the loss is equal to 0. Otherwise the loss is computed in the same way as for the single-operation strategy. The number of credential disclosures is equal to l*j. The number of credential requests is equal to 1.

6.2 Datasets and Experimental Methodology

To make the computation of the loss, the number of credential disclosures, the number of credential requests, the loss ratio, the number of credential disclosures ratio, and number of credential requests ratio independent of the type of Web service transition system, client profile⁶, Web service provider trust and access control policies, and the client behavior ⁷, we have implemented a Java application that:

- randomly generates transition systems;
- randomly generates a client profile;
- creates trust and access control policies;
- randomly simulates client behavior;
- implements the enforcement process of *ACConv*, single-operation and request-all strategies.

We have considered three classes of transition systems. The first class contains transition systems with a number of states varying from five to ten, the second with a number of states varying from fifteen to twenty, and the third with a number of states varying from twenty to thirty. For each class of transition systems, the Java application has created ten transition systems. Then, for each transition system, the Java application has simulated the enforcement process of ACConv, single-operation, and request-all strategies, and has computed the loss, the number of credential disclosures, the number of credential requests, and the loss ratio, the number of credential disclosures ratio, and the number of credential requests ratio, as described in details in what follows.

(1) Transition System Generation. The Java application receives as input the number of states, the number of transitions, and the number of final states that a transition system must have. To generate a transition system, it builds a adjacency matrix having a number of columns and rows equal to the number of states given in input. Then, the application fills in a number of matrix' entries equal to the number of transitions given as input; the position of each transition is randomly assigned.

 $^{^{6}\}mathrm{Here}$ by client profile we mean the set of credentials that a client is willing to disclose to the Web service provider.

⁷The client behavior is modeled as the operations invoked by the client during the interaction with a Web service.

- (2) *Meaningful Conversation Computation*. The Java application computes the set of meaningful conversations associated with the initial state of the generated transition system.
- (3) *Reference Set of Credentials Generation.* The Java application creates a reference set of credentials. Such set of credentials has cardinality equal to the maximum number of transitions in the generated transition system. Such number of credentials allows the generation of access control policies such that each policy specifies conditions on different types of credentials. A sensitivity level is associated with each credential in the reference set. The sensitivity of the credentials is then used to compute the sensitivity of the access control policies that specify conditions on such credentials.
- (4) *Client Profile Generation*. The Java application creates a client profile by randomly choosing a set of credentials from the reference set.
- (5) Trust and Access Control Policies Generation. The Java application has to generate an access control policy for each operation labeling a transition in the Web service transition system. Each access control policy op_i if $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_l, i = 1, \ldots, h$ is such that each term $\mathcal{T}_k, k = 1, \ldots, h$ is an attribute condition $Cond = \mathcal{T}ype : name_{\mathcal{A}} \text{ op } v$ and $\mathcal{T}ype$ is different for each $k = 1, \ldots, h$, where h is a randomly chosen number between 0 and 10. h represents the number of different credential types on which each policy specifies conditions. Once the operation access control policies are generated, the Java application computes the access control policies for the meaningful conversations associated with the transition system's initial state. Then, it computes the sensitivity of the conversation access control policies based on the sensitivity of the credentials on which the policies specify conditions and groups the policies based on their sensitivity. Then, for each set of policies, the Java application defines a trust policy using the same approach used to generate operation access control policies.
- (6) *Client Behavior Simulation.* The Java application simulates the choice of the client by randomly selecting a conversation from the set of meaningful conversations associated with the initial state of the transition system.
- (7) *Enforcement Process Simulation*. The Java application simulates the execution of the *ACConv*, single-operation and request-all strategies' enforcement process. Then, the Java application calculates the loss, the number of credential disclosures, and the number of credentials associated with the conversation chosen to simulate the behavior of the client for the three different access control strategies. Then, for each strategy, the Java application calculates the average value of loss, the number of credential disclosures, and the number of credential disclosures,

6.3 Experimental Results

Figure 4(a) shows the average of the loss ratio values for the three classes of transition systems. The average loss ratio varies in the range [0.2, 0.45] with the number of states and transitions characterizing the classes of transition system we have considered. This means that the loss associated with the *ACConv* strategy is 30% of the loss associated with the single-operation strategy, which has the maximum loss. Figure 4(b) illustrates how the number of credentials ratio varies with the class of transition systems considered. The number of credentials ratio values fall in the range [0.3, 0.6]. Therefore, the number of credentials requested from a user in the *ACConv* strategy is still lower than the number of credential requests by the request-all strategy that maximizes the number of credentials a user is asked to

ACM Transactions on the Web, Vol. 5, No. 3, Article 13, Publication date: July 2011.







Fig. 4. Average of loss ratio and number of credentials ratio values.



Fig. 5. Average of number of credential requests ratio values.

provide. Finally, Figures 5(a) and (b) report the average of the number of credential requests ratios for the three classes of transition systems. The experimental results show that the number of times a user is asked to provide credentials in the *ACConv* strategy represents a good trade-off between the request-all strategy that minimizes the number of credential requests and the single-operation strategy that maximizes it.

We notice that the values of loss ratio, the number of credentials ratio, and number of credential requests ratio increase with the number of transitions. Such an increase is motivated by the fact that with the increase of the number of transitions, the client can perform longer conversations and to complete the execution of these conversations, the client might be entrusted with more than one set of conversation access control policies, and be asked to provide the credentials to satisfy such policies. Thus, the loss associated with each conversations increases, and so the number of credentials disclosed by the client and the number of times the client is requested to provide credentials.

Despite this, the results obtained for ACConv are satisfactory. The first class of transition systems, that represents, according to a recent survey [Benatallah et al. 2003], the most common real Web services, is characterized by a loss that is less than 25% of the loss for the single-operation strategy. Moreover, the number of credentials a user is asked to provide by ACConv is only 30% of the number of credentials for the request-all strategy. Finally, the number of credentials is 20% the number of credentials for the single-operation strategy that is characterized by the highest number of credential requests.

7. RELATED WORK

The problem of disclosure of access control policies has been investigated in Seamons et al. [2001], Yu et al. [2003], Koshutanski and Massacci [2007], and Frikken et al. [2006]. Seamons et al. [2001] and Yu et al. [2003] assume that access control policies can contain sensitive information that should be protected from inappropriate access by strangers during negotiation. They distinguish between policy disclosure and policy satisfaction which allows one to control when a policy can be disclosed from when a policy is satisfied. Koshutanski and Massacci [2007] propose a negotiation scheme for access rights based on two kinds of security policies, access control policies and disclosure control policies. Access control policies state which credentials a client must satisfy to be authorized for a particular resource, while disclosure control policies state which credentials among those occurring in the access control policies are disclosable to (i.e., can be asked to) the client. Frikken et al. [2006] propose a privacy-preserving trust negotiation solution under which clients and service providers are able to learn whether trust can be established without either party revealing to the other party anything about their own private credentials and policies. In ACConv we support an approach to protect access control policies similar to one proposed in Seamons et al. [2001] and Koshutanski and Massacci [2007]. We use two different kinds of security policies: access control policies and trust policies. Access control policies state the credentials a client has to provide to be granted the execution of a conversation. Trust policies state the credentials that must be requested from a client in order to only disclose to the client the set of access control policies associated with the conversations the client is interested in performing.

With respect to access control for Web services, several policy-driven access control models have been proposed, but none of them has investigated access control enforcement for conversational Web services besides *ACConv*. The most relevant approaches in this area are by OASIS with the WS-XACML profile for Web services [Anderson 2007], Wonohoesodo and Tari [2004], Sirer and Wang [2002], Bhatti et al. [2004], Emig et al. [2007], Bertino et al. [2006], Olson et al. [2006], Kagal et al. [2004], Denker et al. [2003], and Agarwal et al. [2004].

XACML consists of an XML-based language for expressing access control policies and a request/response protocol that specifies how to determine if a given action is allowed or not and how to interpret the result. Recently, the XACML Technical Committee has proposed a Web Service Profile of XACML (WS-XACML) [Anderson 2007]

ACM Transactions on the Web, Vol. 5, No. 3, Article 13, Publication date: July 2011.

that specifies how to use XACML in Web services environment. WS-XACML introduces two new types of policy assertion to allow Web service providers and consumers to specify their authorization, access control, and privacy requirements and capabilities regarding Web services interactions. Moreover, WS-XACML proposes an approach to verify whether client's capabilities and Web service provider's requirements match and vice versa. The *ACConv* enforcement process is similar to the WS-XACML process for matching client's capabilities and Web service provider's requirements. In *ACConv* we verify that a client has all the credentials to be granted the execution of the conversations the client is interested in.

Wonohoesodo and Tari [2004] propose SWS-RBAC and CWS-RBAC, two RBAC (Role-Based Access Control) models, respectively, for single and composite Web services. Web services providers' access control requirements are represented as permissions that can be defined both at service and service parameters level. Permissions associates access modes, such as read, modify, execute, write, with parameters. A role is associated with a list of services that it has the permission to execute and with a list of permitted access modes on the service parameters. Therefore, a user in order to access a service must have been assigned to a role that must be granted permission to the service and that has, at least, the minimum access modes on the service parameters. In CWS-RBAC, clients must nominate a global role before they can execute composite services which are mapped onto local roles of the component Web services. The main difference between *ACConv* and the access control models of Wonohoesodo and Tari [2004] is the type of access control enforcement: *ACConv* is a credential-based access control model, while SWS-RBAC and CWS-RBAC are RBAC models.

Sirer and Wang [2002] propose an approach for formally specifying and automatically enforcing security policies. Security policies are specified using a language called WebGuard based on temporal logics. They propose an enforcement engine that automates the task of converting security policies into access control code specific to a particular platform. The enforcement code is integrated in the Web service code and is executed when a Web service's invocation starts and ends. With respect to the *ACConv* policy language, WebGuard is more expressive because it allows one to express access control requirements as temporal conditions on actions executed by clients in the past or that they will execute in the future. However, WebGuard does not provide any mechanism to protect access control policies disclosure while in *ACConv* the disclosure of access control policies is ruled by trust policies.

Bhatti et al. [2004] propose X-GTRBAC, a trust-based, context-aware, role-based access control model for Web services with temporal extensions. X-GTRBAC incorporates the features of the NIST RBAC model and extends it with the specification of temporal constraints related to role, thus enabling user-to-role assignment, permission-to-role assignment, and role activation. Moreover, in X-GTRBAC the assignment of a client to a role is based on the use of distributed TM credentials. Access control policies specify which role is entitled to invoke a given service and temporal and nontemporal contextual constraints. An aspect in common with *ACConv* is related to client identification that is based on identity attributes.

Emig et al. [2007] propose an access control model that is a combination of traditional hierarchical role-based (RBAC) and attribute-based (ABAC) access control models. From ABAC it inherits the way service consumers are authenticated: a consumer is identified by a set of attributes. From RBAC it inherits the definition of hierarchy of roles and the definition of policies as a set of permissions. An access control policy is a combination of permissions combining an object, that is an operation or the whole Web service, and a set of attributes that the consumer has to provide with constraints that take into account the object-associated input parameters and the environment state (like date, time, or any other attribute related to neither the client nor object). Unlike traditional RBAC models, the permissions are not associated with a role but with a set of client's attributes. Moreover, a role does not identify a business role but a set of client's attributes. The main differences with *ACConv* are that access control policies in the model by Emig et al. [2007] allow one to express conditions not only on client credentials but also on contextual parameters, such as date and time, and that the access control enforcement is not flexible. If clients cannot be assigned to a role on the basis of the attributes they present, they are denied access to the service. In *ACConv*, when a client is denied access, the client is requested to provide additional credentials to be assigned to a new set of conversation access control policies.

Bertino et al. [2006] propose WS-AC₁, an access control model with flexible protection object granularity and negotiation capabilities. WS-AC₁ is based on the specification of policies stating conditions on the values of the identity attributes and service parameters that a client must provide to invoke the service. Conditions may also be specified against context parameters, such as time. Further, it is possible to define finegrained policies by associating them with a specific service as well as coarse-grained policies, to be applied to a class of services. The negotiation capabilities of WS-AC₁ are related to both identity attributes and service parameters. Through a negotiation process, the client is driven toward an access request compliant with the service description and policies. Both WS-AC₁ and ACConv are credential-based access control models and are characterized by a flexible policy enforcement process.

Olson et al. [2006] propose an authorization service for Web services based on trust negotiations. In trust negotiation, access policies for resources in the system are written as declarative specifications of the attributes that authorized users must possess. Entities in the system possess digital credentials issued by third parties that attest their attributes, along with policies that limit access to their sensitive resources and credentials. The trust negotiation process allows the parties to mutually disclose their credentials and policies that incrementally establish trust. In order for a client to be granted access to a Web service, the client has to carry out a successful trust negotiation with the authorization service that protects the invocations for that particular Web service. The access control enforcement of ACConv and Olson et al.'s [2006] model are both based on an incremental disclosure of policies and credentials between clients and service providers. However, the model of Olson et al. [2006] does not protect the disclosure of access control policies, while ACConv does.

Finally, other interesting proposals are related to Semantic Web services. The approaches by Denker et al. [2003] and Agarwal et al. [2004] propose to extend OWL-S service descriptions with ontologies that represent Web service provider authorization policies and privacy policies. Both types of policies are expressed in the Rei language [Kagal 2002], which is an RDF Schema-based language for policy specification. It is based on deontic concepts of rights, prohibitions, obligations, and dispensations. These constructs have four attributes: actor, action, provision, and constraint. A constraint specifies some conditions over the actor, action, and any other context entity that must be true at invocation, whereas provision describes conditions that should be true after invocation. Provisions are the actor's obligations.

Agarwal et al. [2004] propose a credential-based access control model for Semantic Web services. They adopt DAML-S [Ankolekar 2002] to represent Web service descriptions in a machine interpretable manner. A DAML-S description has three main parts: ServiceProfile, ServiceModel, and ServiceGrounding. ServiceProfile provides a highlevel description of a service and its provider; ServiceModel details both the control structure and data flow structure of the process represented by the service; Service-Grounding specifies how a service can be accessed. Moreover, Agarwal et al. [2004] adopt SPKI/SDSI credentials [Clarke et al. 2001] to identify the service consumers to which access control policies apply. Access control policies express a set of permissions

ACM Transactions on the Web, Vol. 5, No. 3, Article 13, Publication date: July 2011.

and their temporal validity. They are published in the DAML-S description of the service as preconditions of the process described int the service model part. As access control policies are public, a client can compute the set of SPKI/SDSI credentials to prove its eligibility to access a service.

The approaches by Agarwal et al. [2004] and Denker et al. [2003] have some aspects in common with *ACConv*. First, service consumers are identified by means of credentials that contain attributes. Second, the enhanced service descriptions used in those approaches specify not only the functionalities offered by a service but also their behavioral semantic as in *ACConv*, where the behavioral semantics is represented by a transition system. The main difference is that in those approaches access control policies are all public, while in *ACConv* the disclosure of access control policies is protected by trust policies.

8. CONCLUSIONS

In this article, we presented ACConv, a novel approach for access control enforcement in conversation-based Web services. Our focus is on access control enforcement and limited disclosure of access control policies. While most existing approaches assume a single-operation model in which operations are independent from each other, we assume conversational Web services. Our approach strikes a trade-off between limiting the disclosure of access control policies and allowing clients to complete the execution of a conversation they are interested in. Such approach is based on the notion of meaningful conversation; these are conversations which from a given state reaches a final state in the Web service transition system. The conversation access control policies associated with meaningful conversations are grouped in sets of policies having the same sensitivity level. The disclosure of each set of policies is protected by a trust policy. When a client requests an operation, it is associated with a set of access control policies and a set of conversations referred to as allowable conversations. The set of access control policies is the set of policies, the trust policy of which is satisfied by client's credentials. The allowable conversations are conversations which are protected by the set of access control policies the client has been entrusted with. The allowable conversations which start with the execution of an operation chosen by the client and which access control policy is satisfied by the clients' credentials are referred to as granted conversations. Our approach applies to both simple and composite Web services. We implemented ACConv in a software prototype using the Globus toolkit. The experiments have shown that our approach strikes a good trade-off between the request-all and single-operation strategies.

APPENDIX – COMPLETE RUNNING EXAMPLE

We show how ACConv can be applied to the Amazon Flexible Payments Web service⁸ (Amazon FPS for short). The Amazon FPS provides a set of operations that allows clients to accept payments for selling goods or services, raise donations, execute recurring payments, and send payments. We have considered only a subset of the operations provided by the Amazon FPS to illustrate the ACConv enforcement process, that is the service that allows users to make payments and to manage their transactions. A description of these operations is reported in Table I. Figure 6 shows the Amazon FPS transition system with the selected operations. The different labels represent the operations that a client can invoke from any given state. Final states are represented by gray circles.

⁸http://aws.amazon.com/fps

Table I. Selection of Amazon FPS Operations

Operation	Description
GetAccountBalance	returns the current balance of a customer account
GetAccountActivity	retrieves transactions from an account for a given time period
GetTransaction	fetches the details of a transaction
Pay	initiates a transaction to move funds from the buyer to the recipient
Reserve	reserves the transaction amount on the buyer's credit card or
	debit card for later settlement
Settle	charges an amount previously reserved against the buyer's credit card
Refund	refunds a successfully completed payment transaction
Cancel	voids a transaction is in a reserved state or waiting to be processed
FundPrepaid	transfers money to the account balance



Fig. 6. Amazon FPS transition system.

From the initial state S_0 , a client can select the operations GetAccountActivity, GetTransaction, or GetAccountBalance. A client can list the information about the transactions associated with her account for a given time period by invoking GetAccountActivity operation or she can view the information about a specific transaction by executing operation GetTransaction. Once invoked the operation GetTransaction, the client can decide which operation to invoke according to the status of the transaction as follows.

- If the transaction is pending, the client can invoke operation Pay to complete the transaction. The execution of Pay operation can fail (state S_{10} is reached) or it can be successful (state S_{11} is reached).
- If the transaction is reserved, the client can invoke operation Settle to complete the transaction. The execution of Settle operation can fail (state S_{12} is reached) or it can be successful (state S_{13} is reached).
- If the transaction has been successfully completed, the client can invoke the operation Refund to have his/her money back. The execution of Refund operation can fail (state S_{14} is reached) or it can be successful (state S_{15} is reached).
- If the transaction is reserved or pending, the client can invoke the operation Cancel to void the transaction.



Fig. 7. \mathcal{G}^{scc} of Amazon FPS.

From the initial state, the client can also decide to check her account balance by invoking GetAccountBalance, and then make a one-time payment by executing Pay operation or to reserve the amount of the transaction on her credit card by invoking Reserve operation. The execution of Pay and Reserve operations can fail because the client has not enough funds, and, thus, the client can invoke the operation FundPrepaid to add money to her account. If the execution of the Reserve operation is successful, the client can perform the Settle operation to charge his/her credit card of the amount previously reserved.

To illustrate ACConv enforcement process, we focus our attention on the initial state S_0 . Figure 7 shows the graph of the strongly connected components for the Amazon FPS. We exploit this graph to compute meaningful conversations. Each node in the graph is associated with a triple (cardinality, coverage, rank). Note that the graph is quite similar to the original Amazon FPS transition system because it contains only one strongly connected component composed by the states S_2 , S_3 , and S_6 . This strongly connected component is represented by node C_2 . C_2 has one ingoing node and outgoing node, that is, S_2 . This strongly connected component is associated with the following triple of values (3, 4, 5), indicating, respectively, cardinality, coverage, and rank. The cardinality value is equal to 3 because there are three different operations labeling the transitions between states S_2 , S_3 , and S_6 : Reserve, Pay, and FundPrepaid. The coverage is equal to 4 because the shortest conversation going from state S_2 to state S_2 and including all the operations labeling the transitions between states in C_2 is Pay • FundPrepaid • Reserve • FundPrepaid of length 4. Finally, the rank is equal to 5 according to the formula $\operatorname{rank}(C_2) = 1 + \operatorname{coverage}(C_2) + \operatorname{rank}(C_2)$ because the coverage of C_2 is equal to 4 and the rank of its predecessor node C_2 is equal to 0. The set M^{S_0} of meaningful conversations originating from S_0 contains the following conversations (in the case of conversations with nondeterministic operations, the reached state is specified).

 $-conv_7$: GetAccountActivity \bullet GetTransaction \bullet Settle reaching state S_{13}

ACM Transactions on the Web, Vol. 5, No. 3, Article 13, Publication date: July 2011.

 $⁻conv_1$: GetAccountActivity

 $⁻conv_2$: GetAccountActivity • GetTransaction

 $⁻conv_3$: GetAccountActivity \bullet GetTransaction \bullet Refund reaching state S_{15}

 $⁻conv_4$: GetAccountActivity • GetTransaction • Pay reaching state S_{10}

 $⁻conv_5$: GetAccountActivity • GetTransaction • Settle reaching state S_{12}

 $⁻conv_6$: GetAccountActivity • GetTransaction • Cancel

Table II. Operation Access Control Policies

Operation Access Control Policies		
GetAccountBalance if AmazonID, AmazonPaymentAccountNumber		
GetAccountActivity if AmazonID		
GetTransaction if AmazonID		
Pay if CreditCard : Type = Visa \lor CreditCard : Type = MasterCard \lor		
BankAccountInfo, AmazonPaymentAccountNumber		
$\begin{tabular}{lllllllllllllllllllllllllllllllllll$		
BankAccountInfo, AmazonPaymentAccountNumber		
Settle if TRUE		
Refund if CreditCard : Type = Visa \lor CreditCard : Type = MasterCard \lor		
BankAccountInfo, AmazonPaymentAccountNumber		
Cancel if TRUE		
$\begin{tabular}{lllllllllllllllllllllllllllllllllll$		
BankAccountInfo		

```
-conv_8: GetAccountActivity • GetTransaction • Pay reaching state S_{11}
-conv_9: GetAccountActivity \bullet GetTransaction \bullet Refund reaching state S_{14}
-conv_{10}: GetTransaction
-conv_{11}: GetTransaction • Refund reaching state S_{14}
-conv_{12}: GetTransaction • Settle reaching state S_{13}
-conv_{13}: GetTransaction • Cancel
-conv_{14}: GetTransaction • Pay reaching state S_{10}
-conv_{15}: GetTransaction • Settle reaching state S_{14}
-conv_{16}: GetTransaction • Pay reaching state S_{11}
-conv_{17}: GetTransaction • Refund reaching state S_{15}
-conv_{18}: GetAccountBalance
-conv_{19}: GetAccountBalance • Pay reaching state S_5
-conv_{20}: GetAccountBalance • Pay reaching state S_6
-conv_{21}: GetAccountBalance • Reserve reaching state S_4
-conv_{22}: GetAccountBalance • Reserve reaching state S_3
-conv_{23}: GetAccountBalance • Reserve • Settle reaching state S_7
-conv_{24}: GetAccountBalance \bullet Reserve \bullet Settle reaching state S_8
-conv_{25}: GetAccountBalance • Pay • FundPrepaid
-conv_{26}: GetAccountBalance • Reserve • FundPrepaid
-conv_{27}: GetAccountBalance \bullet Pay \bullet FundPrepaid \bullet Reserve reaching state S_3
-conv_{28}: GetAccountBalance \bullet Pay \bullet FundPrepaid \bullet Pay reaching state S_5
-conv<sub>29</sub>: GetAccountBalance • Reserve • FundPrepaid • Pay
-conv<sub>30</sub>: GetAccountBalance • Reserve • FundPrepaid • Reserve
-conv_{31}: GetAccountBalance • Reserve • FundPrepaid • Reserve • Settle reaching
  state S_7
-conv<sub>32</sub>: GetAccountBalance • Reserve • FundPrepaid • Reserve • Settle reaching
  state S_8
 -conv_{33}: GetAccountBalance • Pay • FundPrepaid • Reserve • Settle reaching
  state S_7
-conv<sub>34</sub> : GetAccountBalance • Pay • FundPrepaid • Reserve • Settle reaching
  state S_8
-conv<sub>35</sub>: GetAccountBalance • Pay • FundPrepaid • Reserve • FundPrepaid • Reserve
• Settle reaching state S_8
```

ACM Transactions on the Web, Vol. 5, No. 3, Article 13, Publication date: July 2011.

Policy identifier	Conversation Access Control Policies
pol_1	conv ₁ if AmazonID
pol_2	equal to pol ₁
pol_3	$conv_3$ if $AmazonID$, $CreditCard$: $Type = Visa \lor$
	$CreditCard: Type = MasterCard \lor BankAccountInfo,$
	AmazonPaymentAccountNumber
pol_4	$conv_4$ if $AmazonID$, $CreditCard$: $Type = Visa \lor$
	$CreditCard: Type = MasterCard \lor BankAccountInfo,$
	AmazonPaymentAccountNumber
pol_5	equal to pol ₁
pol_6	equal to pol ₁
pol_7	equal to pol ₁
pol_8	equal to pol ₃
pol_9	equal to pol ₃
pol_{10}	equal to pol ₁
pol_{11}	equal to pol ₃
pol_{12}	equal to pol1
pol_{13}	equal to pol ₁
pol_{14}	equal to pol ₃
pol_{15}	equal to pol_1
pol_{16}	equal to pol ₃
pol_{17}	equal to pol ₃
pol_{18}	$conv_1$ if $AmazonID$, $AmazonPaymentAccountNumber$
$pol_{19}, pol_{20}, pol_{21}, pol_{22},$	equal to pol ₃
$pol_{23}, pol_{24}, pol_{25}, pol_{26}$	
$pol_{27}, pol_{28}, pol_{29}, pol_{30}$	
$pol_{31}, pol_{32}, pol_{33}, pol_{34}$	
$pol_{35}, pol_{36}, pol_{37}$	

Table III Conversation Access Control	Doligiog	

- conv₃₇ :GetAccountBalance • Pay • FundPrepaid • Reserve • FundPrepaid • Reserve
 • Settle reaching state S₇.

Access control policies protecting single operations are listed in Table II. Conversation access control policies of conversations in M^{S_0} are reported in Table III. The service provider groups these access control policies on the basis of the policies' sensitivity. The different sets of access control policies and the trust policies regulating their disclosure to clients are reported in Table IV. Now suppose that the client being in state S_0 invokes the operation GetAccountBalance and provides her *eBayGold* credit card number proving that she is a rewarded eBay customer. The credential presented by the client is evaluated against the trust policies associated with $C_{l_2}^{S_0}$, the conversation access control policy set that protects conversations $conv_{18}$, $conv_{19}$, $conv_{20}$, $conv_{21}$, $conv_{22}$, $conv_{23}$, $conv_{24}$, $conv_{25}$, $conv_{26}$, $conv_{27}$, $conv_{28}$, $conv_{29}$, $conv_{30}$, $conv_{31}$, $conv_{32}$, $conv_{34}$, $conv_{35}$, $conv_{36}$, $conv_{37}$ that start with the execution of GetAccountBalance, the operation chosen by the client. The client satisfies the trust policy protecting $C_{l_2}^{S_0}$, and as a consequence is asked to provide the credentials *AmazonID*, her *Social Security Number*, and one credential among *CreditCard*,

F. Paci et al.

Access Control Policy Sets	Trust Policy
$Cl_1^{S_0}:\{pol_1,pol_2,pol_5,$	$Cl_1^{S_0}$ if $PicureID: Age > 21$
pol_6, pol_7, pol_{10}	
$pol_{12}, pol_{13}, pol_{15}$	
pol_{18} }	
$Cl_2^{S_0}:\{pol_3,pol_4,pol_8,$	$Cl_2^{s_0}$ if $eBayGoldCreditCard \lor$
$pol_9, pol_{11}, pol_{14}, pol_{16},$	SSN
$pol_{17}, pol_{19}, pol_{20}, pol_{21}$	
$pol_{22}, pol_{23}, pol_{24}, pol_{25}$	
$pol_{26}, pol_{27}, pol_{28}, pol_{29}$	
$pol_{30}, pol_{31}, pol_{32}, pol_{33}$	
$pol_{34}, pol_{35}, pol_{36}, pol_{37}$ }	

Table IV. Sets of Conversation Access Control Policies and Trust Policies Protecting Them

BankAccountInfo and AmazonPaymentsAccountNumber that are requested in the conversation access control policies in set $C_{l_0}^{S_0}$.

ACKNOWLEDGMENTS

The authors would like to thank Simone Savi and Alfonso Calciano for their contributions to the implementation of the software prototype.

REFERENCES

- AGARWAL, S., SPRICK, B., AND WORTMANN, S. 2004. Credential based access control for semantic web services. In *Proceedings of Semantic Web Services*. AAAI. http://www.daml.ecs.soton.ac.uk/SSS-SWS04/Papers.html.
- ANDERSON, A. 2007. Web services profile of XACML (WS-XACML), version 1.0, OASIS standard specification. http://www.oasis-open.org/committees/download.php/24951/ xacml-3.0-profile-webservices-spec-v1-wd-10-en.pdf.
- ANKOLEKAR, A. 2002. DAML-S: Web service description for the semantic web. In *Proceedings of the International Semantic Web Conference (ISWC)*. Lecture Notes in Computer Science.
- BENATALLAH, B., CASATI, F., TOUMANI, F., AND HAMADI, R. 2003. Conceptual modeling of Web service conversations. In Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAISE). Lecture Notes in Computer Science, vol. 2681. Springer.
- BERARDI, D., CALVANESE, D., DE GIACOMO, G., LENZERINI, M., AND MECELLA, M. 2005. Automatic service composition based on behavioral descriptions. *Int. J. Coop. Inf. Syst.* 14, 4, 333–376.
- BERTINO, E., SQUICCIARINI, A. C., MARTINO, L., AND PACI, F. 2006. An adaptive access control model for Web services. *Int. J. Web Serv. Res.* 3, 3, 27–60.
- BHATTI, R., BERTINO, E., AND GHAFOOR, A. 2004. A trust-based context-aware access control model for Web-services. In Proceedings of the IEEE International Conference on Web Services (ICWS).
- CLARKE, D. E., ELIEN, J., ELLISON, C. M., FREDETTE, M., MORCOS, A., AND RIVEST, R. L. 2001. Certificate chain discovery in SPKI/SDSI. J. Comput. Secur. 9, 285–322.
- DENKER, G., KAGAL, L., FININ, T., PAOLUCCI, M., AND SYCARA, K. 2003. Security for DAML Web services: Annotation and matchmaking. In *Proceedings of the 2nd International Semantic Web Conference* (*ISWC*). Lecture Notes in Computer Science, vol. 2870. Springer.
- EMIG, C., ABECK, S., BIERMANN, J., BRANDT, F., AND KLARL, H. 2007. An access control metamodel for Web service-oriented architecture. In Proceedings of the 2nd International Conference on Systems and Networks Communications (CSNC).
- FRIKKEN, B., LI, J., AND ATALLAH, M. J. 2006. Trust negotiation with hidden credentials, hidden policies and policies cycles. In Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS).
- GLOBUS. 2011. Globus toolkit. http://www.globus.org/toolkit/.

GONNET, G. H. AND BAEZA-YATES, R. 1991. HandBook of Algorithms and Data Structures. Addison-Wesley.

INTERNET2. 2006. OpenSAML - an open source security assertion language toolkit. http://www.opensaml.org.

- KAGAL, L. 2002. Rei: A policy specification language. http://rei.umbc.edu/.
- KAGAL, L., PAOLUCCI, M., SRINIVASAN, N., DENKER, G., FININ, T., AND SYCARA, K. 2004. Authorization and privacy for semantic Web services. *IEEE Intell. Syst.* 19, 4, 50–56.
- KOSHUTANSKI, H. AND MASSACCI, F. 2007. A negotiation scheme for access rights establishment in autonomic communication. J. Netw. Syst. Manag. 15, 1, 117–136.
- LAWRANCE, K. AND KALER, C. 2006. Web services security: SOAP message security version 1.1, OASIS standard specification. http://www.oasis-open.org/committees/download.php/16790/ wss-v1.1-spec-os-SOAPMessageSecurity.pdf.
- LEE, A. J. AND WINSLETT, M. 2006. Safety and consistency in policy-based authorization systems. In Proceedings of the 13th ACM Conference on Computer and Communications Security.
- LI, J. AND LI, N. 2006. OACerts: Oblivious attribute certificates. *IEEE Trans. Depend. Secur. Comput. 3*, 340–352.
- MECELLA, M., OUZZANI, M., PACI, F., AND BERTINO, E. 2006. Access control enforcement for conversationbased Web services. In *Proceedings of the 15th International World Wide Web Conference (WWW)*. ACM.
- MOSES, T. 2005. Extensible access control markup language (XACML), version 2.0, OASIS standard. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.
- NUUTILA, E. AND SOISALON-SOININEN, E. 1993. On finding the strongly connected components in a directed graph. *Inf. Process. Lett.* 49, 9–14.
- OLSON, L., WINSLETT, M., TONTI, G., SEELEY, N., USZOK, A., AND BRADSHAW, J. 2006. Trust negotiation as an authorization service for Web services. In *Proceedings of ICDE Workshops*.
- SALTZER, J. H. AND SCHROEDER, M. D. 1974. The protection of information in computer systems. Comm. ACM, 340–352.
- SEAMONS, K., WINSLETT, M., AND YU, T. 2001. Limiting the disclosure of access control policies during automated trust negotiations. In Proceedings of the Conference on Network and Distributing System Security (NDSS'01).
- SIRER, E. G. AND WANG, K. 2002. An access control language for Web services. In Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT). ACM.
- STIRLING, C. 1996. Modal and temporal logics for processes. In Proceedings of the Logics for Concurrency. Structure versus Automata (Eighth Banff Higher Order Workshop). F. Moller and G. M. Birtwistle Eds. Lecture Notes in Computer Science, vol. 1043. Springer.
- SUN. 2003. Sun's XACML implementation. http://sunxacml.sourceforge.net.
- TARJAN, R. E. 1972. Depth-First search and linear graph algorithms. SIAM J. Comput. 1, 146-160
- WONOHOESODO, R. AND TARI, Z. 2004. A role based access control for Web services. In Proceedings of the IEEE International Conference on Services Computing (SCC). IEEE.
- YU, T., WINSLETT, M., AND SEAMONS, K. 2003. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. ACM Trans. Inf. Syst. Secur. 6, 1–42.

Received April 2007; revised July 2010; accepted November 2010