

Instance-Level Security Management in Web Service Business Processes

Dacheng Zhang
University of Leeds
dcz@comp.leeds.ac.uk

Jianxin Li, Jinpeng Huai
Beihang University
{lijx, huaijp}@act.buaa.edu.cn

Abstract

By using Web services, people can generate flexible business processes whose activities are scattered across different organizations, with the services carrying out the activities bound at run-time. We refer to an execution of a Web service based automatic business process as a business session (multi-party session). A business session consists of multiple Web service instances which are called session partners. Here, we refer to a Web service instance as being a stateful execution of the Web service. In [8], we investigate the security issues related to business sessions, and demonstrate that security mechanisms are needed at the instance level to help session partners generate a reasonable trust relationship. To achieve this objective, an instance-level authentication mechanism is proposed. Experimental systems are integrated with both the GT4 and CROWN Grid infrastructures, and comprehensive experimentation is conducted to evaluate our authentication mechanism. Additionally, we design a policy-based authorization mechanism based on our instance-level authentication mechanism to further support trustworthy and flexible collaboration among session partners involved in the same business session. This mechanism allows an instance invoker to dynamically assign fine-grained access control policies for the new invoked instance so as to grant other session partners the necessary permissions.

1. Introduction

A business process is a collection of related structured activities undertaken by organizations in pursuit of certain business goals. When considering the automatic generation of business processes whose activities and data are separated by organisational boundaries, the heterogeneity of software components and legacy systems is an essential issue, and needs to be carefully investigated. In the past decades, much research has been put in identifying techniques and methodologies to effectively integrate heterogeneous software components and reuse legacy software systems; Web services are the latest attempts in this research field. Basically, Web services are an evolution of traditional Web applications. Through the use of Web service technologies, people can create complex applications at run time by integrating loosely coupled, heterogeneous, reusable software components. However, the flexibility and the peer-to-peer style of Web service business processes also introduce new challenges to security systems. For instance, two competitive business sessions may have activities undertaken by instances spawned by the same Web service. Although these instances are potentially competitive, they are executed on the same computer and share same resources,

e.g. data, memory, etc. Moreover, because the session partners of a business session may be controlled by different organizations, the business session management system cannot monitor and manage its session partners as effectively as in the conventional business process instances. All these issues make business sessions vulnerable to malicious attacks. In this paper we present our efforts in generating security boundaries for business sessions. Our approach achieves authentication and authorization for business sessions at the instance level. The authentication system can generate and verify the identities of session partners at run-time. In comparison with conventional authorization approaches for business processes, our authorization system is more flexible as it permits session partners to dynamically assign the access control policies to the instance they invoked. Moreover, with the assistance of our authentication system, the authorization system can enforce the business session constraint which specifies that an instance can get the access to another instance only when the two instances are involved within the same multi-party session. In Section 2, we discuss instance-level authentication issues for web services and introduce our authentication system. In Section 3, we show our experimental systems which are used to evaluate the overhead introduced by the authentication

system. Experimental results are also presented and analysed. In Section 4, we briefly discuss the limitations of existing authorization systems in business process management, and then introduce our instance-level authorization system. Finally, Section 5 concludes the paper.

2. Instance-Level Authentication

Authentication is always a fundamental concern when a security infrastructure for distributed systems is designed. In order to validate the identities of the principals in distributed systems, authentication systems are employed. Without losing generality we regard an automatic business process as an automatic distributed system, and the principals working within a business session are Web service instances. In this section, we discuss the authentication issues for Web service instances within business sessions and propose an instance-level authentication mechanism.

2.1 Authentication Issues in Web Service Business Sessions

When a Web service receives an initial request from a business session, it normally generates a new instance to deal with this request. This instance is then involved within the business session and may also deal with other requests from its session partners. Consider that the session partners may be generated and bound at run-time, and cooperate in a peer-to-peer way [1]. Moreover, the specification of the communication amongst the session partners on some occasions cannot be precisely defined in advance. Thus, there are cases where a service instance has to deal with requests from other instances which it never communicated before [8]. When an instance receives such a request, the instance needs an authentication mechanism to help it verify the identity of the sender in case that the request is sent from an instance in other business session by mistake, or from a malicious attacker.

Essentially, there are two goals for an authentication system [7]. One is to establish the identities of the principals; another is to distribute secret keys for principals. Such secret keys can be used in further communication amongst the principals or used to generate new short-term secrets. In conventional authentication mechanisms (e.g., PKI, Kerberos, etc.), the identities of the users and the secret (e.g., password) used in the authentication processes are generated and deployed in

advance. For example, in PKI, the certificate authority can generate a public key pair and sign a certificate for every user. The public key pair is forwarded to the user out-of-band, and the certificate is stored in a directory server and published over the network. Thus users can use the certificates and the key pairs obtained in advance to prove their identities during the communication with other users. In contrast to these classical authentication mechanisms, instance-level authentication systems have to generate and distribute both the identities of the instances and the secrets used to prove the identities at run-time as the instances are dynamically generated.

Most current Web service systems have not carefully considered authentication issues in business sessions yet. In GT4, service instances can be identified by their resource keys, but the solutions to proving the possession of the identifiers are not standardized. We can use user certificates to generate proxy credentials at run-time for every new generated instance, and thus verify the identities of service instances. However, the proxy certificate solution cannot be directly used to enforce the security boundaries of business sessions, as there is no notion of business session in this solution at all.

2.2 Instance-Level Authentication in [5]

Hada and Maruyama [5] propose a session authentication mechanism that can realize basic authentication functions for session partners. In their protocol a Session Authority (SA) component is introduced to take charge of distributing session authentication messages. The SA generates a session secret for every business session. When a new service instance is invoked and involved within a business session, the associated session secret will be forwarded to the instance, and thus the instance can use this secret to authenticate itself to its session partners. This authentication protocol can effectively authenticate the session partners working within the same business session and prevent the uncontrolled messages transport across the boundaries of business sessions. However, there are still many security issues in the area of business session management which needs to be carefully considered. For instance, Hada and Maruyama's protocol only can distinguish the session partners in one business session from the service instances in other business sessions. However, in some scenarios a fine-grained control over the session partners is required. For example, a Web service may have

multiple instances acting in different roles within the same business session; these instances need to be explicitly distinguished. Moreover, in Hada and Maruyama's protocol, there is only one session secret for every business session; the whole business session will be comprised if this session secret is disclosed. Thus, this protocol is not suitable for business sessions with high security requirements. Based on above discussion, we propose in [8] a fine-grained authentication protocol which is complementary to Hada and Maruyama's protocol. Our protocol explicitly identifies the session partners at the instance level. We also design a new key management mechanism which generates secrets and distributes them to session partners at run time. These secrets can be used in the authentication processes amongst session partners.

2.3 Multi-Party Authentication System for Service Instances

Since Web service instances are the principals of business sessions. Our authentication system attempts to directly authenticate service instances rather than users, which is different from the conventional authentication systems mentioned above. In our authentication system, every instance is associated with a pair of keys generated by using the Diffie-Hellman algorithm. The public key is used as the identifier of the instance, while the private key is kept secretly and used to prove the possession of the identifier. Similar to the solution in [5], there is a SA in our solution. All the identifiers of the session partners and other related information of a business session are stored in the SA. When a new service instance is invoked by a business session, the identifier and other related information will be sent to the SA so as to guarantee the validity of the information kept by the SA. A SA instance is associated with this business session to manage the associated session information. Additionally, the SA instance can provide reliable real-time information to session partners. For example, when an instance receives a request from a stranger, the instance can query the SA instance whether the requester is a session partner.

As mentioned above, an important objective of authentication systems is to generate and distribute secret keys for principals. Because our authentication system is an identity-based mechanism, if two session partners have obtained each other's identifiers they can generate and share a secret key without any

additional communication. In our authentication protocol, Message Authentication Codes (MACs) are used to prove the integrity of the messages transported between session partners and thus verify the origin of a message. A nonce is inserted into the SOAP header of every message to foil the man-in-the-middle attack before messages are sent out. The nonce is generated from a unilaterally increased counter which is held by each Web service instance. When sending a message out, the instance increases its counter. Suppose instance *A* receives a message from instance *B*. If the counter number in the message is smaller than or equal to the counter number in the message sent from *A* before, *B* will reject this message. Interested reader can be referred to [8] for more detailed introduction about our authentication protocols.

3. Experimental System

Figure 1 illustrates the process of invoking a new Web service instance and introducing it to the SA as a session partner. Because the authentication system is implemented on the Grid infrastructures, a Web service is associated with a factory service to manage the resources where the state information is stored. In our experimental system, the identifier of an instance and associated private key is stored within a resource, and the instance identifier is identical to the resource key. In the first step (messages 1 and 2), the invoking instance on service 1 contacts factory service 2 in order to generate a new resource.

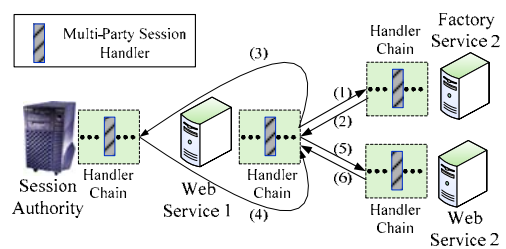


Figure 1: Operations of authentication

In Grid, the state information of Web services is stored in resources. Therefore, we can locate a service instance when the corresponding resource is found. After receiving the reply from factory service 2 and gaining the resource identifier, the invoking instance introduces the new invoked instance to the session authority (messages 3 and 4). If the reply from the SA implies that the new invoked instance has been accepted as a session partner, the invoking instance sends message (message 5) to the new

invoked instance and waits for the response (message 6). Compared with the original instance-invoking process in Grid, two additional messages (messages 3 and 4) are brought by the session authority. Note that message 1 needs to be protected by additional security measures. Because the new resource has not been generated, and thus the MAC associated with the message cannot be calculated. After the resource has been generated, the integrity and freshness of messages 2 ~ 6 can be protected by using MACs and nonces.

We have implemented two experimental systems. The first experimental system (*ES1* for short) consists of a SA service and three experimental services. In this experiment, three experimental services repeatedly invoke each other in the sequence demonstrated in Figure 2 until a particular amount of service instances have been spawned and introduced to the SA.

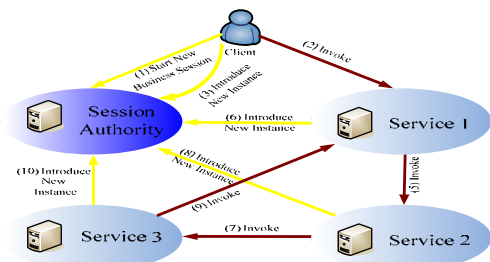


Figure 2: Experimental system with the SA

Compared with *ES1*, the system structure of the second experimental system (*ES2* for short) is relatively simple. *ES2* consists of three experimental services. In the experiments, the experimental services invoke each other repeatedly until a particular amount of service instance has been generated. *ES1* is used to evaluate the performance of the system incorporated with the SA, whilst *ES2* is used to evaluate the performance of the Grid infrastructure. By comparing the experimental results obtained from *ES1* and *ES2*, we can evaluate the influence introduced by our authentication mechanism on the performance and scalability of Web service systems.

3.1 Experiments on a Single Computer

We deploy the experimental systems on a single computer for two reasons. Firstly, in order to precisely evaluate the overhead introduced by our authentication security protocols (e.g., generating key pairs, generate MACs for messages, etc.) we need to remove the influence brought by the time consumed on transporting

messages. Secondly, if an experimental system is deployed on different computers, operations in the system may be executed in a concurrent fashion. Deploying the experimental system can help us to evaluate the performance of the systems in the worst case where all the operations of the system are executed sequentially. In this sub-section all the experiments are deployed on a single computer unless mentioned otherwise.

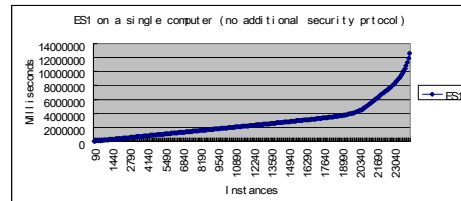


Figure 3: ES1 deployed on GT4

The experiment illustrated in Figure 3 is used to evaluate the scalability of *ES1*. In this experiment, more than 24,000 instances are generated and introduced to the SA. From the beginning of the experiment, the experimental system stays in a stable state. The time consumption of the system is proportional to the number of the generated instances, until over 16,000 instances are generated. After that, the performance of the experiment becomes bad, and the system finally stops due to the lack of the memory. In this experiment, no additional security protocol is used to secure the messages transported between instances. In experiment shown in Figures 4 and 5, we incorporate *ES1* and *ES2* with messages level protocols (i.e., secure conversation and secure message provided by GT4).

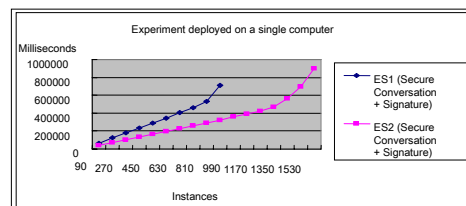


Figure 4: ES1 and ES2 on GT4 + Secure Conversation

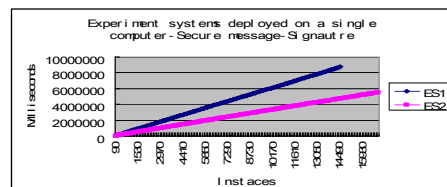


Figure 5: ES1 and ES2 on GT4 + Secure Message

When *ES1* is incorporated with the secure conversation protocol, that is, the secure conversation protocol is used to generate signatures for every message transported in the experiment, the time consumption of the experimental system starts increasing non-linearly after over 720 instances are generated. The same phenomenon occurs in *ES2* after 1040 instances are generated when *ES2* is incorporated with the secure conversation protocol. As illustrated in Figures 4 and 5, the time consumption of *ES1* is about as twice as that of *ES2*, while *ES1* can realize fine-grained authentication at the instance level.

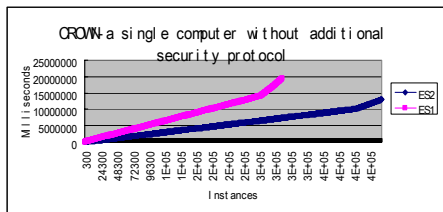


Figure 6: Compare between *ES1* and *ES2* on CROWN

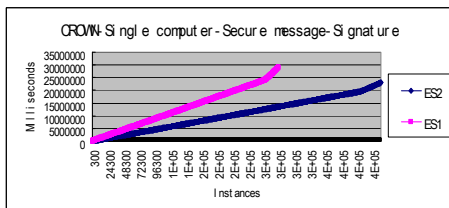


Figure 7: Compare between *ES1* and *ES2* on CROWN + Secure Message

We have successfully implemented our design in the large-scale CROWN (China Research and Development environment Over Wide-area Network) Grid. CROWN aims to promote the utilization of valuable resources and cooperation of researchers nationwide and world-wide. In the experiments presented in Figure 6, our experiments system executed without any additional security protocol. In the experiments presented in Figure 7, our experimental system uses the security message protocol to protect the integrity of the messages transported between the instances generated in the experiments. In these experiments, our authentication system can stay in a stable state until over 260,000 instances are generated. This demonstrates that although the authentication protocol introduces some overheads to the system, the scalability of the authentication system is still very good.

3.2 Distributed Deployed Experiments

After evaluating the performance of the experimental systems deployed on a single computer, we deploy the experimental systems in a distributed way and further evaluate them in a more realistic environment. In the experiments introduced in this sub-section, the experimental systems are distributed unless mentioned otherwise. The scalability of the experimental systems deployed in a distributed way is much better than on a single computer. As illustrated in Figure 8, the time consumption of the distributed *ES1* increases linearly, until more than 70,000 new instances are generated. When incorporated with the secure conversation protocol (see Figure 9), *ES1* executes stably until over 3,000 instances have been generated. In the experiments shown in Figures 10 and 11, the time consumption of *ES1* is proportional to the number of the new generated instances until over 300,000 instances are generated.

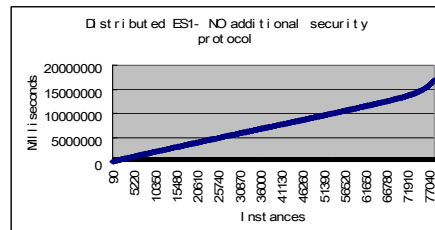


Figure 8: *ES1* deployed on GT4

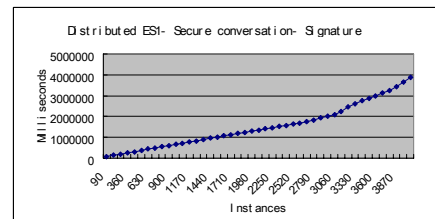


Figure 9: *ES1* deployed on GT4+ Secure Conversation

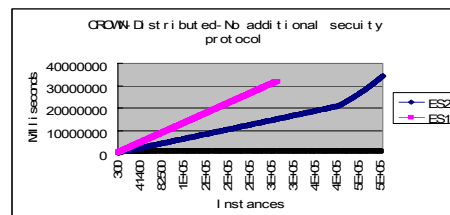


Figure 10: Compare between *ES1* and *ES2* on CROWN

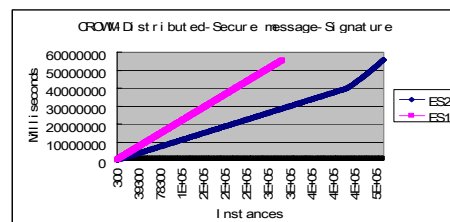


Figure 11: Compare between *ES1* and *ES2* deployed on CROWN + Secure Message

3.3 Analysis

The experimental results presented above demonstrate that our authentication system can realize the instance level authentication with a reasonable cost. Additionally, we conclude that the scalability of our experimental systems is influenced by several factors:

- ❖ The security protocols which our authentication system is incorporated with. When incorporated with different security protocols, the scalability of *ES1* notably varies. For example, when *ES1* is deployed on a single computer and incorporated with the secure conversation protocol, the time consumption increases nonlinearly after over 720 instances are generated. When *ES1* is cooperated with the secure message protocol, the system can execute stably even after generating more than 13,000 instances.
- ❖ The amount of the memory. In *ES1*, the SA stores the information about the session partners in memory, and the resources of experimental services are stored in memory as well. Thus, the amount of the memory is critical to the scalability of *ES1*. When the experimental systems are deployed on a single computer, the SA and the experimental services have to share the limited memory. This is a main reason that the scalability of the distributed experimental systems is much better than the experimental systems deployed on a single computer.
- ❖ The time and memory consumed on reading and writing the log files. In our experiments, all the log information is stored in a log file. As the length of the log file increases, the time and memory consumed on reading and writing the log file increase too.

4. An Instance-Level Authorization Mechanism

Beside instance-level authentication, instance level authorization is another essential requirement to generate secure service-based business processes [8]. Previous research in authorization for business processes rarely considers the scenarios where multiple parties cooperate in a peer-to-peer way, and a lot of issues need to be further explored.

4.1 Limitations of the Current Authorization System in Workflows

Workflow is a popular technology to implement automatic business processes. Despite some research [2, 4] has been done to address the instance level authorization and authorization issues in workflow systems, existing approaches can not be adapted to flexible business processes that consist of a large number of instances which are dynamically bound and collaborated. The workflows considered in these approaches are normally compliant with the workflow reference model [6] proposed by the Workflow Management Coalition. In this model business specifications are executed by a centralized workflow engine, and the participants of workflows are passive. The participants just collect work items from their work lists, achieve jobs, and send results back. All the interactions among instances are managed by the workflow engine. Thus, the access control rules are configured by the administrator statically, and the participant instances have no privilege to dynamically specify their own access control policy. In contrast with the business process supported by the conventional workflow systems, Web service business processes rely on the peer-to-peer interactions between participant instances. This is because Web services are designed for the cross-organizational environments, and have to face the lack of a central location for the middleware [1]. Therefore, the communication between service instances can be very complex, and it is common that multiple instances cooperate to achieve a business goal. For example, in a business session, a session partner invokes a medical data processing service, which can be accessed openly, and generates an instance to processing the session partner's medical data, which, however, only can be accessed by special instances involved in this business session, such as a hospital, thus the invoker needs to assign its own access control policy to this new invoked instance. In order to address this issue, we propose an instance-level authorization mechanism which is designed for Web service business processes. Compared with other authorization solutions for business processes, our authorization system has the following features:

- ❖ The invoker can dynamically specify the access control policies of the invoked instance at run-time, and thus the access

control relationship between instances is more flexible.

- ❖ With the assistance of the instance-level authentication system, our authorization mechanism can define the business session constraints within the access control policies.

4.2 Design of Authorization Mechanism

As Web services are loosely coupled and can be bound at run-time, the number of the session partners involved within a business session may keep changing. Therefore, it is difficult for an instance to obtain real-time knowledge about its session partners when it is involved in a peer-to-peer business session. For instance, when an invoker sets the access control policies to a new invoked instance, it may not know the identifiers of the session partners who are going to access this new instance. On extreme occasions, the instances supposed to access the new instance have not been spawned yet. Therefore, identity-based authorization systems are not suitable, and we propose an attribute-based authorization mechanism instead.

We adopt XACML (eXtensible Access Control Markup Language) to express fine-grained access control policy. As illustrated in Figure 12, our authorization system consists of two components, the authorization service (AuthzService for short) and the SA service.

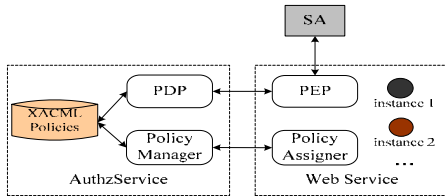


Figure 12: The architecture of the system

The authorizing processes can be broken down to two parts: assigning the access control policies to the new created instance and making authorization decisions with respect to the requests to access instances. During the process of policy assigning, the instance invoker can contact the *Policy Assigner* of the target service to specify access control policies, and then the *Policy Assigner* will call the *Policy Manger* in the AuthzService to write this policy and make it available to the *PDP* (Policy Decision Point). When there is a request for this instance controlled by dedicated policy, The *PEP* (Policy Enforcement Point) of this instance will firstly contact the SA and query the authentication result. Secondly the *PEP* accesses the *PDP* of

the AuthzService; the *PDP* then makes an authorization decision according to this authenticated Token. In practice, the AuthzService can be implemented in various ways. For example, the AuthzService can be implemented as a centralized service which deals with the requests from all the session partners within the business session. The AuthzServices also can be implemented in a decentralized way; that is, each AuthzService only manages the service instances within a local domain, and multiple AuthzServices are associated with a business session.

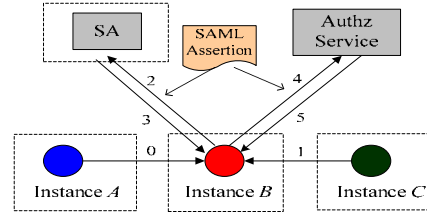


Figure 13: An example of the authorization process

Figure 13 presents an example of the authorization enforcement. In this example there are three instances in a business session, *A*, *B* and *C*. *A* invokes *B*, and specifies an access control policy encoded with XACML for the new generated *B*. *C* also intends to access *B*. Assume that *A* has stored the policy into the authorization service of *B* (step 0 in Figure 13). The steps of the authorization enforcement in Figure 13 are presented as follows:

1. $C \rightarrow B : \{request, id_C, attrs_C\}$. *C* sends the *request* of service access, its instance identifier id_C , and other related attributes information $attrs_C$ to *B*.
2. $B \rightarrow SA : \{auth_{request}\}$. *B* encapsulates *C*'s information into a SAML assertion $auth_{request}$ and sends $auth_{request}$ to the SA to query whether *C* is in the same business session with *B*.
3. $SA \rightarrow B : \{auth_{result}\}$. The SA verifies whether *C* is involved in this business session, and replies a SAML assertion $auth_{result}$ which consists of the authentication result to *B*.
4. $B \rightarrow AuthzService : \{auth_{result}, id_B\}$. After receiving the response from the SA, *B* encapsulates the authentication result and its instance identifier to the AuthzService.
5. $AuthzService \rightarrow B : \{authz_{result}\}$. The AuthzService makes an authorization decision according to the XACML polices

that B assigned in advance and returns the authorization result back.

During the processes of authentication and authorization, SAML (Security Assertion Markup Language) is used to describe related security assertions in the messages. For example, the attributes (e.g., identifier) of an instance ought to be encapsulated according to the SAML specification.

5. Conclusion

Conventionally, business process supporting techniques (e.g., workflow) mainly deal with the issues of implementing business process within an organization, and business processes supported by these techniques are normally implemented in the client/server model. A Web service business process may have to execute in a cross-organizational environment where the client/server model is not suitable. Web services have to achieve business objectives through peer-to-peer cooperative interactions [1, 3]. The structures of Web service business processes thus can be more flexible and complex than conventional business processes. Because session partners cooperate in a peer-to-peer way, a session partner may lack the real-time information about its session partners. All these issues bring new challenges to security systems; a lot of security issues also need to be carefully considered. In this paper, we present our efforts in instance-level authentication and authorization. An instance-level authentication system is proposed. Compared with traditional authentication systems, this system is able to generate the identities for session partners and authenticate them at run-time. Our instance-level authentication system can help a service instance to distinguish its session partners from the instances in other sessions, and thus generate a security boundary for business sessions. However, considering the potential competitions among session partners and a lot of other security requirements (e.g., the separation of duty), we propose an instance-level authorization system so as to further improve the security of business sessions. In most conventional authorization systems for business processes, the access control relation within a business session is statically predefined by the administrator of the business session. In Web service business sessions, session partners works in a peer-to-peers way and their relationship is more flexible and difficult to be precisely predefined. Therefore, our authorization system enables session partners to dynamically assign access control policies for

the new invoked instances and ensure the correctness and the security of the cooperation between the session partners.

We deployed our experimental systems on the GT4 and CROWN Grid infrastructures. These experimental systems are mainly used to evaluate the scalability and the performance of our instance-level authentication system. In some experiments, our authentication system can execute stably until over 260,000 instances are generated. Such experimental results demonstrate the scalability of the instance-level authentication system is good. Additionally, compared with the performance of service-level security mechanisms, the overhead introduced by our instance-level authentication system is reasonable. We are now in the process of evaluating the performance of the instance-level authorization system and going to present the results in the papers published in the future.

Reference

- [1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, "Web Services," *Springer Verlag*, 2003.
- [2] V. Atluri, WK Huang, "An Authorization Model for Workflows," *Proc. the 5th European Symposium on Research in Computer Security*, Lecture Notes in Computer Science, Vol. 1146, Springer-Verlag, pp. 44-64, 1996.
- [3] G. B. Chafle, S. Chandra, V. Mann and M. G. Nanda, "Decentralized Orchestration of Composite Web Services," *Proc. the 13th international World Wide Web conference on Alternate track paper & posters*, pp. 134-143, May 2004.
- [4] D. Domingos, A. Rito-Silva, P. Veiga, "Authorization and Access Control in Adaptive Workflows," *Proc. 8th European Symposium on Research in Computer Security*, pp. 23-38, 2003.
- [5] S. Hada and H. Maruyama, "Session Authentication Protocol for Web Services," *Proc. 2002 Symposium on Application and the Internet*, pp. 158-165, Jan. 2002.
- [6] D. Hollingsworth, "Workflow Management Coalition: The Workflow Reference Model," *Technical Report WPMC-TC-1003*, Workflow Management Coalition, Brussels, Belgium, 1994.
- [7] T. Y. C. Woo and S. S. Lam. "A Semantic Model for Authentication Protocols," *Proc. IEEE Symposium on Research in Security and Privacy*, pp. 178--194, Oakland, California, May 1993.
- [8] D. Zhang and Jie Xu, "Securing Instance-Level Interactions in Web Services," *Proc. 2005 ISADS IEEE*, pp. 443-450, April 2005.