Preserving Privacy in Dynamic Web Service Composition

Dieter Hutter and Melanie Volkamer

German Research Center for Artificial Intelligence (DFKI GmbH)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
[hutter, volkamer]@dfki.de

The proliferation of web services as self-contained web accessible programs and the idea of the Semantic Web of making information *computer-interpretable* enables the dynamic composition of complex services assembled from various individual services and typically distributed over the web. Following the paradigm of pervasive computing, pro-active agents are installed on mobile phones or PDAs operating on the web and handle the context-aware discovery of appropriate services and use AI-based plan generation techniques to dynamically compose the retrieved service to solve complex tasks. Functional composition of complex services is well understood and supported.

However, the introduction of web services in general and dynamic web service composition (e.g. [2, 7]) in particular requires appropriate security facilities to guarantee the security requirements of all participants. On the one hand, web services have to be protected against misuse of their resources, and on the other hand, the user of web services require the privacy of their data. Standard approaches for secure execution of services such as those using REI [6] or Ponder [3] are based on the specification of access control policies to control the individual execution of services and thus focus on the first task. We propose an approach [4, 5] to ensure the privacy of user data by introducing a dynamic security check between plan generation and execution of the plan. Thereby we can guarantee that the execution of a synthesized plan will not distribute user's or newly generated data to a non-entitled web service. We make use of techniques developed in program language security in general and an adapted version of Volpano Smith's [8, 9] security type calculus in particular.

To protect the privacy of user related information, the data used in web services is always *classified* according to its confidentiality. Web services require the corresponding *clearances* to deal with confidential data. Both classifications and clearances are denoted by a so-called *security types*. There is a partial ordering $\leq$ on security types which allow us to compare them. The set of all security types together with $\leq$ forms a lattice. Similar to the approach presented by Bell and LaPadula [1], the idea is that a web service is only entitled to obtain a specific datum if its clearance is at least as high as the classification of the data.

However, in practice we would like to select the clearances of web services and also the classification of data with respect to individual categories or types of information. For instance, while we trust a travel agent to keep our travel routes confidential we might have mixed feelings when providing the same agent with a direct debit authorization for our bank account. Analogously, a given datum may allow us, for instance, to infer confidential information about the location of a person or confidential information about his bank account. Hence, both classifications of data and clearances of web services are described by a vector of security types. Each entry in the vector denotes the classification or clearance with respect to a particular category or roles (like, for instance, location or payment information).

The clearances of web services are assessed by the owner of the planning agent or, in case of a *delegation*, by an agent acting on behalf of the provider. Thus, an information provider (typically the user) can specify which web service it trusts to keep data private with respect to given categories, or not. Analogously, the user

determines the classification of the data that he enters to the agent and that will be provided to web services during plan execution. Web services classify data they provide as an output by integrating the classifications of all data used to compute the result. The data can be either input data from the caller or own data. In the first case the classification is given by the caller and in the second case the requirements of their own security policies for this data has to be taken into account.

The set of categories used to invoke a web service may change while processing the request. When a called web service provides new information, it may introduce a new category and classify the new data also with respect to this new category. In this way a subsequently called web service can formulate its security requirements for its provided data by defining also the clearances of all web services with respect to the new category. In this case, all data provided by the calling web service have to be classified as public with respect to the newly introduced categories. This is to avoid the blocking of external data by classifying them as confidential for a new category but providing no web service with a corresponding clearance.

All the above mechanisms are embedded in the extended type calculus. The extension belongs to the call of web services. In this context, we defined *security specifications* for web services which are defined and published by the agents similiar to the functional specifications. Thus, the information flow of confidential data is monitored by the type calculus which propagates the security requirements of the user and of the involved web services along the planning and execution of dynamically composed web services.

The described mechanisms together with the type system prevent any kind of "accidental" revealment because the plans are checked before the execution. However, the user of a planning agent has to trust that the web service will not send confidential data to some unauthorised web services. To overcome this problem we expand our approach by the application of certified web services. To ensure that only the certified parts of an agent get knowledge of the received input the approach has to be extended by the application of Trusted Computing or Proof Carry Code mechanisms.

References

1. D. E. Bell and L. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Technical Report MTR-2997, MITRE, 1976.
2. J. Brison, D. Martin, S.I. McIlraith, and L.A. Stein. Agent-based composite services in DAML-S: The behavior-oriented design of an intelligent semantic web. 2002.
3. Naranker Dulay, Nicodemos Damianou, Emil Lupu, and Morris Sloman. A policy language for the management of distributed agents. In *Agent Oriented Software Engineering (AOSE-2001)*, pages 84–100. Springer, LNCS 2222, 2001.
4. Dieter Hutter and Melanie Volkamer. Information flow control to secure dynamic web-service composition. In *3nd International Confernce on Security in Pervasive Computing*. Springer, LNCS, 2006.
5. Dieter Hutter, Melanie Volkamer, Matthias Klusch, and Andreas Gerber. Provably secure execution of composed semantic web services. In *International AAMAS-Workshop on Privacy and Security in Agent-based Collaborative Environments, PSACE-2006*, Hakodate, Japan, 2006.
6. Anand Patwardhan, Vlad Korolev, Lalana Kagal, and Anupam Joshi. Enforcing policies in pervasive environments. In *Mobile and Ubiquitous Systems, MobiQuitous-2004*, pages 299–308. IEEE Computer Society, 2004.
7. M. Sheshagiri, M. desJardins, and T. Finin. A planner for composing services described in DAML-S. In *Proceedings of AAMAS 2003 Workshop on Web Services and Agent-Based Engineering*, 2003.
8. Dennis M. Volpano and Geoffrey Smith. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(3):167–187, 1996.
9. Dennis M. Volpano and Geoffrey Smith. A type-based approach to program security. In *TAPSOFT'97: Theory and Practice of Software Development*, pages 607–621. Springer, LNCS 1214, 1997.