Service Integration towards Security Orchestration

Aradhana Goutam, Raj Kamal, and Maya Ingle

***Abstract*—Integrating applications is not simple. The code, which is often distributed between client and server, is highly error-prone and complicated to retain. In this paper, we introduce a generative environment for abstract services and non functional characteristics. The mediation code is necessary to make applications interoperate. This executes all the operations that are necessary for true communication between two applications. The paper describes the functions required for the service integration towards security orchestration.**

***Index Terms*—Orchestration, service integration, security.**

I. Introduction

The software system plays very important role more and more into our company and many views of our life. The new services help become us and used becomes in masses [1]. These new services are interoperating with the applications in general indebted. These services integrate around the business data and rational firm processes arrange themselves. For example, in telecommunications, the new services on the inherited applications are required for network direction, invoicing systems and managing customer direction, etc.

The service composition is complex for several reasons. First, there are a lot of technologies to describe, the edition and to compose the services. The Web Services are related to a lot of norms in evolution. Nowadays, the service composition cannot be based on the service specifications only. Syntactic Compatibility does not guarantee semantic compatibility. In practice, the service composition is based on the hard encoded rules that allow attaining the foreseen results. A service composition has to attain a series of pre-defined non functional qualities, as the security for example, which demands the production of complicated code.

This code is often distributed between client/server and difficult to maintain. In this parchment, we present a tool that supports Orchestration and security. This tool furnishes an extensible solution to express separately manage flows of non functional property. The Orchestrator Model for System Security (OSS) is designed for service integration with security orchestration.

The parchment is organized as it follows. First, somebasic conditions for integration and support. Section III is proposal based on an approach models motivated. The ePO and OSS are compared in Section IV, Section V presents the orchestration activities and Service Integration is presented in Section V. Section VI concludes this parchment.

Manuscript received March 4, 2012; revised April 20, 2012.

Aradhana Goutam is with the School of Computer sc. and Information Technology, Devi Ahilya University, Indore, India (e-mail: aradhana.pande@gmail.com).

Raj Kamal and Maya Ingle are with School of Computer Science and Information Technology, Devi Ahilya University, Indore, India.

II. Classification of Security Services in Web Services

Service integration depends on public services. Service selection is according to user requirement, and concrete run-time behavior of services. Security makes service integration harder. The run-time permissions depend on a suitable abstraction of the history of all the pieces of code (possibly partially) executed so far. This approach has been receiving major attention, at both levels of foundations [17, 18, and 19].

The security activities i.e. reading and writing files, service invocation, are called events. Sequences of events are called histories. The security classification is on following aspects:

Stateless / Stateful Services:

A service which does not keep histories is stateless service. A service which keeps the history is stateful service [20].

Local / Global Histories:

Locally generated events are known as local histories. Global history may cross over multiple services [20].

Dependent / Independent Threads:

An independent thread keeps separate history, while dependent threads may share part of their histories. Therefore, dependent threads may influence each other when using the same service, while independent threads cannot. [20]

Service is an interpreted as functional type, of the form $S1 \text{--}^{RH}\text{-->} S2$. This represents the relationship between two services i.e.S1evaluates to an object of type S2. The annotation ^{RH} is a history expression. The ^{RH} directs the selection of services that respect the requested properties about security or other non-functional aspects.

Service interfaces can implement using call-by-policy primitive. For instance, service integration can be mechanically inferred through a type and effect system. A policy P is a integration of services. To select a service matching a given policy P, and with functional type $S1 \rightarrow S2$, a client request for Req ($S1 \text{--}^{P}\text{-->}2$). The call-by-Policy ensures the service selection, with respect to policy P.

A call-by-policy is a standard service call, and it is formalized as a mapping from requests to services.

III. Non-Functional Properties

Integrating applications descend not simply to the simple

calls of service. The mediation code is necessary to make applications interoperate.

Communication is primary goal of mediation to enable the applications that use the different communication protocols to interoperate. This is executed by means of the protocol transformation as in a network bridge [5].

- Syntactic alliance function is to align data formats. This can be done between every application or by an intermediary format. [5].
- Semantic alliance function is to align semantic data. This is required because many of the applications have major differences in the definitions of function and of behaviors [5].
- Non functional property is to guarantee the certain property of quality in the application exchanges, for example the security or availability [5].
- Storage is to take annotations of all the exchanges between the applications. The mediation layer can't provide support for service call, responses and the data [5].
- Interception is to collect system check data that verify the foreseen quality of service [5].
- Business logic codes: The mediation layer can be used to insert logic code, as an access to a database for instances. This approach is useful but its usage is not recommended [5].

The mediation layer can be used through an EAI (Enterprise Application Integration). EAI is an integration framework composed of different technologies and services. This is a middleware to enable integration of systems and applications across the enterprise [1]. Web services aim for the solutions of lighter integration and established the definition of Enterprise Service Buses, or ESB. ESB [2] software is used for design and implementation. It is also used for interaction and communication between mutually interacting software applications in SOA. The Services of client and Web components requires mediation for communication. These components execute the previously presented operations (the protocol transformation, Syntactic alliance and semantic, the insert of non functional code, etc.). It provides a unique interface for all the different elements (i.e. the applications) implied in a communication. This reduced the number of point to point connections. Fig. 1 shows the service integration through ESB.

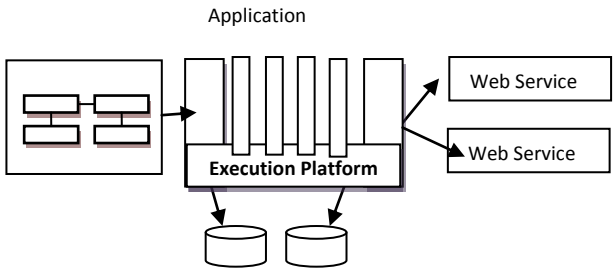

Fig. 1. Service integration through ESB.

IV. Proposal

Generative approach for OSS is based on two drivers: Abstraction and Separation. OSS model having an abstracted service defined in the following terms:

- The functional interfaces specify the functional characteristics of services.
- The property, identified by their names and types.

Abstracts service can be composed and executed with a physical service. A composite abstract service is described as a simple abstracted service and executed by a series of physical services.

A physical service is an implementation: it is furnished with the code (the files). It is illustrated with interfaces and property. The interfaces can be the Java interface, IDL-affection interfaces, etc.

Fig. 2 shows the service integration in OSS. It is clear from the figure that single security policy is composed of different integrated service. These integrated services are the sub-set of the security policy. These security policies are the sub-set of the physical services.

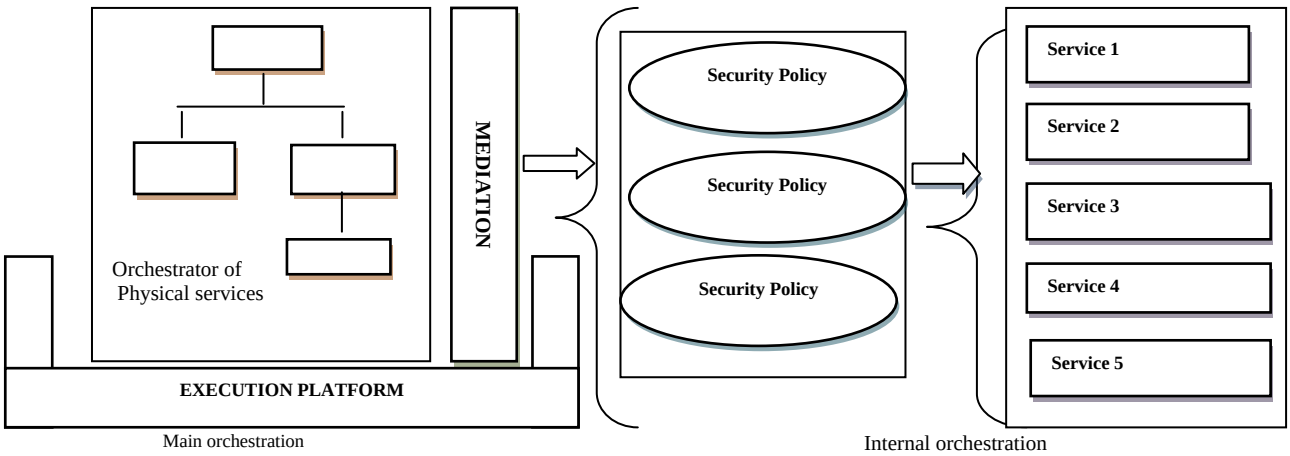

Fig. 2. Service integration in OSS.

The mediation layer controls the communication between the security policies and orchestrator. There are two layered

orchestration in the system. Internal orchestration is at security policy i.e. each security policy orchestrate different service according to the service call either parallel or sequential.

- Main orchestration is physical service orchestrator/ security policy orchestrator i.e. different security policies get selected according to security aspect.

V. Orchestrator Model for System Security (OSS) and McAfee ePolicy Orchestrator

ePO is the security management software for systems, networks, data, and compliance solutions. The following are the flaws in ePO:

- It works for the large business organizations only.
- Identity Management is not integrated.
- It's very expensive and hard to manage for small scale organization.
- Feasibility of adding several non-functional properties is not possible.

The above mentioned flaws are improved in Orchestrator model for system security (OSS). The following are the improvements or addition of new facilities in OSS:

- The feasibility of addition of several non functional properties.
- The principle of the separation is used to simplify the system. Advantage of this separation is that the models can be easily maintained.
- The dynamic selection of services allows weak coupling of property.

The flaws ePO are improved in proposed OSS. Service Integration with security orchestration is taken into account in OSS. It makes easy service selection, dynamic selection of security services and two level orchestrations are possible.

The service integration with security orchestration is introduced in proposed OSS. This reduces the loading time and selection time because of the separation concept.

Merits of Service Integration with OSS:

- Service selection is easy.
- Dynamic selection of security services.
- Two level orchestrations are possible.
- Separation of module makes easy composition.

Demerits of Service Integration with OSS:

- Less scaling options, harder to incrementally grow the platform as traffic or users increase.
- The CPU requirements reach peak levels simultaneously, performance, and stability may be adversely affected.
- IP multicast design require, because service selection for multi-user affect CPU load.

The service integration with security orchestration reduces the loading time and selection time because of the separation concept.

VI. Orchestration Activities

Security orchestration enables the services in trusted area. Fig. 3 shows an OSS framework [10]. Functioning of the OSS is as follows: All services are called through a gateway as per security policy. A request message (for example 3 for authentication) is given to an Orchestrator. The Orchestrator invokes the security services in a response message as per the request 3). An exception is thrown to the Orchestrator, if a service check fails [YM05]. The gateway and Orchestrator function as a single entity.

OSS contains the filters and security services. Example of security services are Authentication, Validation and Authorization. Consider following example. It explains the working principle of OSS. User can access the available service only if request passes through the OSS.

User sends request to access service 2 (Step 0). This will invoke public registry of service 2. A message will be generated. This message is passed to the Orchestrator (Step 1). The security- services are executed either in parallel or sequential.

Once a process completes through the steps 2, 3, and 4, 5, 6, 7, 8, 9 the Orchestrator will respond to user through step 10. If any intermediate step fails then an exception is thrown. The Orchestrator will stop the processing. It will send a service deny message to the user.

Table I lists Orchestrator activities as a function of time. The Orchestrator selects the security activities at different instants either parallel or sequential. Instances are assumed as T1, T2, T3….and are such that T1<T2< T3…. .

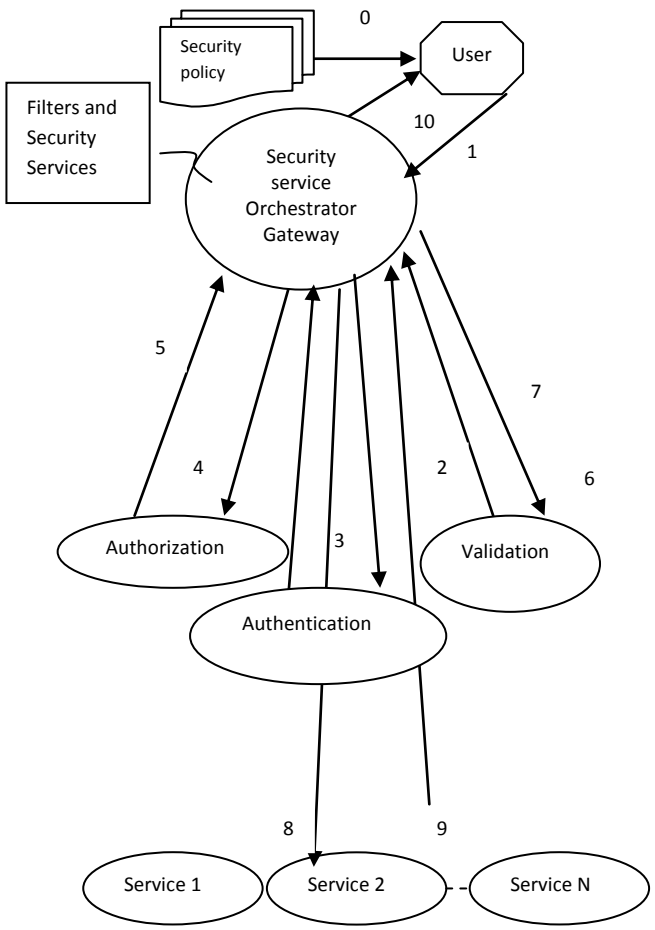

Fig. 3. Basic OSS framework.

VII. INTEGRATION OF SERVICES

The integration of security service and Orchestrator models is required. A model for execution is as follows:

The selection of the services takes into consideration in model for execution. An Execution model maintains the state of the execution environment. Application simplifies an intranet management for security services using OSS. The model for execution allows relation between service properties and service dynamism.

The Execution model is like a directory. It understands the information of available physical services. It also takes into account historic invocations of the services. The content of these directories is as follows:

TABLE I: ACTIVITY TIME FOR SECURITY SERVICES ORCHESTRATOR

Activity	Orchestration	Sub-Activity	Sub-Activity Orchestration	Start Instance	Finish Instance
Authentication	Parallel	Username	Parallel	T1	T2
		Signature	Parallel	T1	T3
	Sequential	X.509 Certificate	Sequential	T3+ΔT3	T4
		Encryption	Sequential	T4+ΔT4	T5
Integrity	Parallel	Signature	Parallel	T5	T6
	Sequential	X.509 Certificate	Sequential	T6+ΔT6	T7
		Encryption	Sequential	T7+ΔT7	T8
Confidentiality	Sequential	Confidentiality	Sequential	T8	T9
Audit (Logging)	Sequential	File	Parallel	T9	T10
		Database	Parallel	T9	T11
Service Element Selection	Sequential/ Parallel	----------	-------	T11’, T11’’, T11’’’	T12, T12’, T12’’
Execution	Sequential/ Parallel	----------	----------	T13	T14

Approach in the execution phase is as follows:

1) A selection mechanism allows choosing most appropriate service.
2) A physical service modelling is done according to the type of service. The model takes into account the services functionality and security characteristics.

3) Code generation process.

VIII. Conclusion

Proposed OSS model supports the addition of the security in the service composition during SOC. Service Integration allows two level orchestration i.e. internal and external orchestration. Service selection is easy. Dynamic selection of security services is possible. Separation of module makes easy composition. Security service Orchestration becomes flexible.

OSS tool easily integrates itself in the system of information of a business. It suffices to adapt the access to the system of identity management according to the used technology. It is easy to add any number of non-functional properties.

An application of the OSS is described. The results of an experiment performed are described. A chain of data from a data acquisition system is used in the experiment. The experiment shows implementation of security aspect using the OSS.

Inter-sector and inter-enterprise working will be more important in future. Because, the mutual technologies are increasing day-by day and offer low-cost, efficient means of coordination. OSS can be added more flexibility and made easily extendible for inter-enterprise.

REFERENCES

[1] Wikipedia. [Online]. Available: http://En.Wikipedia.Org.

[2] IBM. Web Services Atomic Transaction (WS-AtomicTransaction). (November 2004). [Online]. Available: http://www.download.boulder.ibm.com/ibmdl/pub/software/dw/library/ws-atomictransaction 200411.pdf.

[3] D. A. Chappell, *Enterprise Service Bus*, O'Reilly Media Inc., 2004.

[4] E. Pulier and H. Taylor, "Understanding Enterprise SOA," *Manning*, 2006.

[5] S. Chollet and P. Lalanda, "An Extensible Abstract Service Orchestration Framework," in *Proc. the 2009 IEEE International Conference on Web Services*, pp. 831-838.

[6] M. P. Papazoglou and D. Georgakopoulos, "Service-Oriented computing: Introduction," *Commununications of the ACM*, vol. 46, no. 10, pp. 24–28, October 2003.

[7] G. Pedraza and J. Estublier. "An Extensible Services Orchestration Framework through Concern Composition," *International Workshop on Non-functional System Properties in Domain Specific Modeling Languages*, 2008.

[8] Apache. (2006). Web Services Security for Java. [Online]. Available: http://www.ws.apache.org/wss4j/.

[9] M. Colombo, E. D. Nitto, and M. Mauri, "SCENE: A Service Composition Execution Environment Supporting Dynamic Changes Disciplined Through Rules," in *Proc. International Conference on Service Oriented Computing*, pp. 191–202, 2006.

[10] C. Blundo, E. De Cristofaro, C. Galdi, and G. Persiano, "Validating Orchestration of Web Services with BPEL and Aggregate Signatures," in *Proc. the 2008 Sixth European Conference on Web Services*, pp. 205-214.

[11] M. Bartoletti, P. Degano, and G. L. Ferrari, "Types and Effects for Secure Service Orchestration," in *Proc. the 19th IEEE workshop on Computer Security Foundations*, USA: IEEE Computer Society Washington, DC, 2006.

[12] M. Bartoletti, P. Degano, and G. L. Ferrari, "Types and Effects for Secure Service Orchestration," in *Proc. the 19th IEEE workshop on Computer Security Foundations*, USA: IEEE Computer Society Washington, DC, 2006.

[13] S. Chollet and P. Lalanda, "An Extensible Abstract Service Orchestration Framework," in *Proc. the 2009 IEEE International Conference on Web Services*, pp. 831-838.

[14] M. D. D. Fabro, J. Bézivin, and P. Valduriez, "Weaving Models with the Eclipse AMW plugin," presented at Eclipse Modeling Symposium, Eclipse Summit Europe 2006, Esslingen, Germany, 2006.

[15] Fundamental Security Concepts. [Online]. Available: http://www.mhprofessional.com/downloads/products/0072254238/0072254238_ch01.pdf .

[16] G. Pedraza and J. Estublier, "Distributed Orchestration versus Choreography:The FOCAS Approach," presented at ICSP International Conference on Software Process, *LNCS*, Vancouver Canada: Springer Verlag, May 16-17.

[17] A. Banerjee and D. A. Naumann, "History-based acce ss control and secure information flow," in G. Barthe, L. Burdy, M. Huisman, J.-L. Lanet, T. Muntean, (eds.), *Cassis 2004 LNCS*, vol. 3362, Heidelberg: Springer, 2005.

[18] P. W. Fong, "Access control by tracking shallow execution history," *IEEE Symposium on Security and Privacy*, 2004.

[19] C. Skalka and S. Smith, "History effects and verification," in W.-N. Chin, (ed.), *APLAS 2004 LNCS*, vol. 3302, Heidelberg: Springer, 2004.

[20] M. Bartoletti, P. Degano, G. L. Ferrari, and R. Zunino, *Secure Service Orchestration*, Berlin Heidelberg: Springer-Verlag, pp. 24–74, 2007.

[21] P. W. Fong, "Access control by tracking shallow execution history," *IEEE Symposium on Security and Privacy*, 2004.

[22] A. Lapadula, R. Pugliese, and F. Tiezzi, "A calculus for orchestration of web services," *European Symposium in Programming Languages*, 2007.

[23] A. Lazovik, M. Aiello, and R. Gennari, "Encoding requests to web service compositions as constraints," in: P. van Beek, (ed.), *CP 2005, LNCS*, vol. 3709, Heidelberg: Springer, 2005.

[24] J. Misra, "A programming model for the orchestration of web services," presented at 2nd International Conference on Software Engineering and Formal Methods, 2004.

[25] M. Bartoletti, P. Degano, and G. L.Ferrari, "Planning and verifying service composition," Technical Report TR-07-02, Dip. Informatica, Univ. of Pisa.

[26] M. Bartoletti, P. Degano, and G. L. Ferrari, "Enforcing secure service composition," in *Proc. 18th Computer Security Foundations Workshop*, 2005.

[27] A. Banerjee and D. A. Naumann, "History-based access control and secure information flow," in G. Barthe, L. Burdy, M. Huisman, J.-L. Lanet, and T. Muntean, (eds.), *Cassis 2004. LNCS*, vol. 3362, Heidelberg: Springer, 2005.

[28] B. Atkinson *et al.*. (2002). Web Services Security (WS-Security). [Online]. Available: http://www.oasis-open.org

[29] K. Bhargavan, C. Fournet, and A D. Gordon, "Verified reference implementations of WS-security protocols," in M. Bravetti, M. N ú˜ nez, and G. Zavattaro, (eds.), *WS-FM 2006. LNCS*, vol. 4184, Heidelberg: Springer, 2006.

[30] D. Gorla, M. Hennessy, and V. Sassone, "Security policies as membranes in systems for global computing," *Logical Methods in Computer Science*, vol. 1, no. 3, 2005.

Aradhana Goutam is pursuing Ph.D. Devi Ahilya Vishwavidyalaya, in Indore (M.P.). She is an assistant professor of the Department of Information Technology, Father Conceicao Rodrigues College of Engineering, Bandra (West), Mumbai. She has totally 7 years of experience.

Raj Kamal was born in 1949. He obtained his Ph.D. in 1972, in IIT Delhi. He is a senior-most Professor in Devi Ahilya University School of Computer Sciences, Electronics and Information Technology. He has totally 43 years of teaching and research experience. Former-Vice Chancellor Total 9½ Months (Indore, 2006-07 and 2009). He has published totally 109 research papers in journals and conferences of both international and national repute.

He has twelve text books (refer www.rajkamal.org) for students of Computer, Electronics, Communication and Information Technology, which includes books on Embedded Systems and Internet and Web technologies published from McGraw-Hill India, McGraw-Hill China, McGraw-Hill South Korea, McGraw-Hill Singapore, McGraw-Hill U.S.A. Computer Architecture Schaum Series, Mobile Computing Oxford Univ. Press and Microcontroller- Pearson Education.

Maya Ingle is a professor of SCSIT, DAVV, Indore. He obtained his Ph.D of Computer Science in Devi Ahilya University, Indore (M.P.) India, M.Tech (CS) Indian Institute of Technology, Kharagpur, India, Post Graduate Diploma in Automatic Computation, University of Roorkee, Roorkee, India, and M.Sc. (Statistics) Devi Ahilya University, Indore (M.P.) India. He has around 80 research papers which have been published in International/ National Journals and Conferences. Around 26 years of Technical and Administrative experience. Major Research Areas of Interest is Software Engineering, Statistical Natural Language Processing, Usability Engineering, Agile computing, Natural Language Processing, Theoretical Computing, Algorithms, Object Oriented Software Engineering.