# Dynamic Access Control Policies and Web-Service Composition

F. Siewe, H. Janicke, K. Jones

{fsiewe, heljanic, kij}@dmu.ac.uk
Software Technology Research Laboratory,
Gateway House, De Montfort University,
Leicester LE1 9BH

**Abstract.** Service composition is a fundamental technique for developing web-service applications. In general, a single service is not enough to achieve the user's goal, rather several services, often from different providers, are composed dynamically to satisfy a request. Ensuring security in such a system is challenging and not supported by most of the security frameworks proposed in current literature. This paper presents a formal model for composing security policies dynamically to cope with changes in requirements or occurrences of events. The model can be used to specify the security policies of web-services and to reason about their composition. We illustrate our approach with a simple example from healthcare services.

## 1 Introduction

Service Oriented Computing (SOC) is gaining prominence as the need for dynamic discovery and binding, is becoming an important requirement for internet based business applications. A particular standard-based instantiation of Service Oriented Computing, *Web services*, is seen by many as a viable platform for integrating scientific as well as business applications that operate in distributed and heterogeneous environments [1,2,3]. In this setting service composition is the main mechanism for synthesizing answers to users' requests. It is important to be able to reason about the security properties of such a composition in order to ensure security in web-service applications.

WS-Security [4] describes enhancements to SOAP messaging to provide quality of protection through message integrity, confidentiality, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies. WS-Policy [5] provides a general purpose model and corresponding syntax to describe and communicate the policies of a web-service. WS-SecurityPolicy [6], built on the WS-Policy and the WS-PolicyAssertion [7], is a declarative XML format for programming how web-service implementations construct and check WS-Security [4] headers. None of these models provides constructs for expressing the composition of the security policies attached to web-services.

This paper presents a formal framework for expressing dynamic security policies and their composition. Central to our objective is the specification and verification of security policies that can change dynamically to cope with timeliness and occurrences of events. In this respect we base our approach on Interval Temporal Logic (ITL). ITL is a flexible notation for both propositional and first order reasoning about periods of time found in descriptions of hardware and software systems. Unlike other temporal logic, ITL can handle both sequential and parallel composition [8]. We develop a security language based on this formalism. A rich set of operators is defined for expressing security policies driven by events or time [9]. The advantage of using ITL is that the functional and temporal requirements can be specified in the same formalism. As a consequence authorisations based on the history of execution and the state of the system can be specified and enforced. Furthermore, one can reason about the functional, temporal and security properties of systems in a uniform manner.

The rest of the paper is organised as follows. We first present a short informal introduction to ITL, and then go on to introduce our dynamic security policy model in section 3. A small example taken from the health-care environment shows how the policy model is applied. The paper then concludes and outlines future work in the area.

## 2 Interval Temporal Logic

Interval Temporal Logic (ITL) is a linear-time temporal logic with a discrete model of time. A system is modelled by a set of relevant state variables. An interval is considered to be a (in)finite nonempty sequence of states $\sigma_0\sigma_1\ldots$, where a state $\sigma_i$ is a mapping from the set of variables to the set of values. The length $|\sigma|$ of a finite interval $\sigma$ is equal to the number of states in the interval minus one. An empty interval has exactly one state and its length is equal to 0.

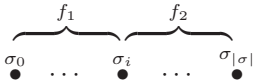
The syntax of ITL is defined as follows, where $a$ is a static variable (does not change within an interval), $A$ is a state variable (can change within an interval), $v$ a static or state variable, $g$ is a function symbol, and $p$ is a predicate symbol.

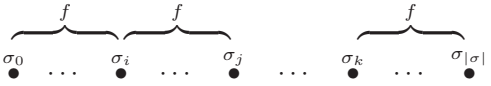
- Expression: $exp \ ::= \ a \mid A \mid g(exp_1, \ldots, exp_n) \mid \iota\, a : f$
- Formulae: $f \ ::= \ p(exp_1, \ldots, exp_n) \mid \neg f \mid f_1 \wedge f_2 \mid \forall v \cdot f \mid skip \mid f_1; f_2 \mid f^*$

The function symbols include e.g. arithmetic operators such as $+, -$ and $*$ (multiplication). A constant is denoted by a function without parameter, e.g. $2, 3$ or 5. An expression of the form $\iota\, a : f$ is called a *temporal expression*. It returns a value $a$ for which the formula $f$ holds in the reference interval. Atomic formulae are constructed using relation symbols such as $=$ and $\leq$. Formulae are then constructed by composing atomic formulae with first order connectives (e.g. $\neg, \wedge, \forall$) and the temporal modalities *skip* (i.e. an interval of exactly two states), ";" (*chop*) and "*" (*chopstar*). Due to the space limitations, the readers are referred to [8] for the formal semantics and the proof system of the logic.

An informal semantics of the temporal modalities can be defined as follows:

– The formula *skip* denotes a unit interval (length equal to 1), i.e.
  $skip$ : 

– The formula $f_1; f_2$ holds for an interval if the interval can be decomposed ("chopped") into a prefix and a suffix interval, such that $f_1$ holds over the prefix and $f_2$ holds over the suffix, or if the interval is infinite and $f_1$ holds for that interval, i.e.
  $f_1; f_2$ : 

– Finally the formula $f^*$ holds for an interval if the interval is decomposable into finite number of intervals such that for each of them $f$ holds, or the interval is infinite and can be decomposed into an infinite number of finite intervals for which $f$ holds, i.e.
  $f^*$ : 


A formula with no temporal operators in it is called a *state formula*. Such a formula holds for an interval if it holds in the first state of the interval.

## 3 A Dynamic Access Control Model

Access control policies are expressed in terms of subjects, objects and actions. Subjects represent active entities, such as users and processes, that can be authenticated within the system. The system state is represented by objects. Objects can only be modified by the execution of actions on request of authenticated subjects. The access control policy determines whether a subject is allowed to perform an action on an object, or not.

In the context of web-services a service is seen as a resource that is provided within the system, to which access is controlled. A service can also request other services and is actively involved in computation. In our formal policy model, a web-service can therefore be seen as both object and subject. The type of request made to the web-service is modelled as an action.

Traditionally access control policies are defined in terms of rules that capture access control requirements [10]. The general form of a rule is:

*premise* $\longrightarrow$ *consequence*

The premise of a rule determines when the rule *fires* and the consequence of the rule determines the outcome of the rule, for example an access control decision. We follow this approach, but allow the premise of a rule to express a behaviour rather than a predicate. The intuition is that an authorisation can be dependent on the history of execution rather than only the currently observable state. This allows the expression of history dependent authorisations such as the Chinese Wall Policy [11].

The informal semantics of operator $\mapsto$ (*Followed By*), that is used in the rules is: Whenever $f$ holds for a subinterval, $w$ holds in the last state of that

subinterval.The right-hand side of a rule in the security model contains either the variable $autho$, $autho^+$ or $autho^-$. This allows to express hybrid access control policies, in which both positive authorisation ($autho^+$) and negative authorisation ($autho^-$) can be expressed. In case of conflicts, i.e. a subject has both positive and negative authorisation, a conflict resolution rule ($autho$) determines the actual access decision. Eq. 1 shows a conflict resolution rule, stating that a negative authorisation takes precedence over a positive authorisation.

$$autho^+(S,O,A) \wedge \neg autho^-(S,O,A) \mapsto autho(S,O,A) \tag{1}$$

We denote universally quanitified variables by uppercase letters. Rules form the basis of our access control model. A simple policy can be seen as a set of these rules, where the intuition is that all rules apply simultaneously. To capture the dynamics of certain security requirements and to allow the incremental development of security policies, policies can be composed using a rich set of operators, described by the following BNF:

$P := p \mid P\,;Q \mid P^\frown Q \mid \langle w\rangle P \mid [w]P \mid t : P \mid P^* \mid P^\oplus \mid P \rhd^t_w Q : R \mid P \| Q$

Where $p$ denotes a simple policy, $w$ denotes a state formula, $t$ a natural number, $P, Q$ and $R$ range over policies. The informal description of the operators follows:

- $P\,;Q$: Sequential composition of two policies. The system is first governed by policy $P$ and then by policy $Q$.
- $P^\frown Q$: Like ";", although between P and Q lies a unit interval.
- $\langle w\rangle P$: The system is governed by policy $P$ unless $w$ holds. The state formula $w$ can here indicate the happening of an event.
- $[w]P$: The system is governed by policy $P$ as long as $w$ holds.
- $t : P$ ; defines that policy $P$ is enforced for $t$ time-units.
- $P^+$ : defines an iteration of policy $P$.
- $P^\oplus$ : Like $P^+$ with one unit-interval between the repetitions.
- $P \rhd^t_w Q : R$ : Behaves like policy $P$ until a condition $w$ becomes true or $t$ time-units elapse. It then changes to behave like $R$ if $w$ is true or like $Q$ otherwise.
- $P \| Q$ : Both, policy $P$ and policy $Q$ apply at the same time.

Policy composition can be used for the incremental development of security policies. The advantage of this approach is that small policies are easier to comprehend and verify. The compositional operators can then be used for the integration of the overall system policy. For further details and a formal semantic of the policy language we refer the reader to [9]. The following section shows the incremental development of a policy for a simplified health-care scenario.

## 4 Case Study

Access to electronic patient records underlies sophisticated restrictions, which are often dependent on external events. Although, we restrict ourselves here to

a few requirements, more complex policies can be specified using the model. A more comprehensive list of requirements can be found for example in [12].

**Scenario:** *Patients can always access their own medical records, but not append any information. The clinician that created a medical record is responsible for that record. The responsible clinician can read and append to records he/she is responsible for. In case of a national emergency situation (e.g. epedemic disease) the protection of personal information stored in health-records is relaxed. To ensure that a sufficient service can be provided, all health-care professionals are allowed to read information stored on electronic patient records.*

The requirements are formalised in the following authorisation rules.

$$\texttt{owner}(P, R) \mapsto \mathit{autho}^{+}(P, R, \texttt{read}) \tag{2}$$

$$\texttt{owner}(P, R) \mapsto \mathit{autho}^{-}(P, R, \texttt{append}) \tag{3}$$

$$\Diamond \mathit{do}(C, R, \texttt{create}) \mapsto \mathit{autho}^{+}(C, R, \texttt{read}) \tag{4}$$

$$\Diamond \mathit{do}(C, R, \texttt{create}) \mapsto \mathit{autho}^{+}(C, R, \texttt{append}) \tag{5}$$

$$\texttt{role}(C, \texttt{hcProf}) \mapsto \mathit{autho}^{+}(C, R, \texttt{read}) \tag{6}$$

Rules (2) and (3) express that the owner $P$ of a medical record $R$, indicated by the predicate $\texttt{owner}(\texttt{P}, \texttt{R})$, can `read` his/her record, but not `append`. The rules (4) and (5) capture that The responsible clinician $C$ can `read` and `append` to records $R$ that he/she created. ($C$ is responsible, if he/she created the record in the past.) Finally rule (6) defines that in an emergency situation, health-care professionals can read information stored in electronic patient records.

The rules ((2)...(6)) are the basic elements that compose the overall security policy. Let $P = \{(2), (3)\}$ now be the simple policy that applies normally, $Q = \{(6)\}$ be the simple policy that applies in under emergency conditions, $R = \{(1), (4), (5)\}$ the simple policy that applies over the whole policy composition. $R$ defines the conflict resolution and that the responsible clinician can access the record. The latter is necessary, to ensure that the temporal reference in the rules applies over the whole policy composition.

$$S = R \parallel (\langle \texttt{emergency}() \rangle P \,;\, [\texttt{emergency}()](P \parallel Q))^{+} \tag{7}$$

The policy composition provides a flexible way to express complex requirements in an modular fashion. The example shows how a dynamic security policy governs the access to resources provided by a single service. The composition of policies becomes more important when web-services with different security policies are composed. Our security model can be used to describe this change and to reason about properties the composition exhibits.

## 5 Conclusion and Future Work

We have shown how security and functionality within a system should be considered as contemporaneous requirements. Access permissions may change as a

result of the execution of a given service or based upon the system state. This requires a security model in which one can express temporal aspects. With this in mind, we have proposed a dynamic, policy-based access control model. Policies are based upon Interval Temporal Logic (ITL) and are expressed in terms of subjects, objects and actions. Such policies are dynamic in the sense that they can be temporally composed in response to events or time constraints. This has been shown with a small scenario taken from the health-care environment.

Future work includes the extension of tool support for the specification and analysis of policies. We are also developing linguistic support in form of the design language SANTA for secure *multi-agent* systems, which is utilising our dynamic security policy model to express application-level access control.

## References


1. Solanki, M., Cau, A., Zedan, H.: Augmenting Semantic Web Service Description with Compositional Specification, WWW2004. In: International Conference on World Wide Web, New York, USA (2004) 544–552 1
2. Automating DAML-S Services Composition Using SHOP2. In: 2nd International Semantic Web Conference. (2003) 1
3. Mukhi, N., Plebani, P.: Supporting Policy-driven Behaviors in Web Services: Experiences and Issues. In: International Conference on Service Oriented Computing, New York, USA (2004) 322–328 1
4. Nadalin, A., Kaler, C., Hallam-Baker, P., Monzillo, R.: OASIS Web Services Security: SOAP Message Security 1.0. (WS-Security) 1
5. Box, D., Curbera, F., Hondo, M., Kaler, C., Langworthy, D., Nadalin, A., Nagaratnam, J.N., Nottingham, M., von Riegen, C., Shewchuk, J.: Web Services Policy Framework. (WS-Policy) 1
6. Della-Libera, G., Hallam-Baker, P., Hondo, M., Janczuk, T., Kaler, C., Maruyama, H., Nagaratnam, J.N., Nash, A., Philpott, R., Prafullchandra, H., Shewchuk, J., amd R. Zolfonoon, E.W.: Web Services Security Policy Language. (WS-PolicyAssertions) 1
7. Box, D., Hondo, M., Kaler, C., Maruyama, H., Nadalin, A., Nagaratnam, J.N., Patrick, P., von Riegen, C., Shewchuk, J.: Web Services Policy Assertions Language. (WS-PolicyAssertions) 1
8. Moszkowski, B.: Compositional reasoning using interval temporal logic and tempura. In Roever, W.P.d., Langmaack, H., Pnueli, A., eds.: Compositionality: The Significant Difference. Volume 1486 of LNCS., Berlin, Springer Verlag (1998) 439–464 1, 2
9. Siewe, F., Cau, A., Zedan, H.: A Compositional Framework for Access Control Policies Enforcement. In Backes, M., Basin, D., Waidner, M., eds.: ACM Workshop on Formal Methods in Security Engineering (FMSE'03), Washington, DC, ACM Press (2003) 32–42 1, 3
10. Woo, T.Y.C., Lam, S.S.: Authorization in distributed systems: A formal approach. In: Proceedings of the 13th IEEE Symposium on Research in security and Privacy, Oakland, California, May 4-6 (1992) 33–50 3
11. Brewer, D., Nash, M.: The Chinese Wall Policy. In: IEEE Symposium on Research in Security and Privacy. (1989) 206–214 3
12. Anderson, R.: Security in clinical information systems. Technical report, BMA Report, British Medical Association (1996) 4