A POLICY-BASED APPROACH TO SECURE CONTEXT IN A WEB SERVICES ENVIRONMENT

Zakaria Maamar$^{\alpha}$, Ghita Kouadri Mostéfaoui$^{\beta}$ and Djamal Benslimane$^{\delta}$
$^{\alpha}$*Zayed University, Dubai, U.A.E.*
$^{\beta}$*University of Montreal, Montreal, Canada*
$^{\delta}$*Claude Bernard Lyon 1 University, Lyon, France*

Keywords: Context, Policy, Security, Web service.

Abstract: This paper presents a policy-based approach for securing the contexts associated with Web services, users, and computing resources. Users interact with Web services for personalization needs, and Web services interact with resources for performance needs. To authorize any context change, a security context is developed. The security context reports on the strategies that protect a context using authorization and restriction policies.

1 INTRODUCTION

Over the last three years, we have been investigating *Web services* along the following directions: context-aware composition, personalization, self-coordination, semantic mediation, and last but not least tracking (Benslimane et al., 2005; Maamar et al., 2006a). This paper continues our investigation of Web services personalization (Maamar et al., 2005) with emphasis this time on securing the content of the contexts of the components that participate in such a personalization.

User, *Web service*, and *resource* represent these components. Users trigger Web services to satisfy their needs, and Web services operate on top of resources during computing. To track the progress of interaction that happen between users and web service, and then between Web services and resources, specific structures of type *context* are devised (Fig. 1). In (Maamar et al., 2005), we specialized context into user context ($\mathcal{U}$-context), Web service context ($\mathcal{W}$-context), and resource context ($\mathcal{R}$-context). Coutaz et al. argue that "*context is not simply the state of a predefined environment with a fixed set of interaction resources. It is part of a process of interacting with an ever-changing environment composed of reconfigurable, migratory, distributed, and multiscale resources*" (Coutaz et al., 2005).

It is widely agreed that any tracking operation relies on the quality of information that is collected, refined, and used for feeding this operation. A poor quality of information usually results in making wrong decisions, which affect for example the chronology of operations to execute, the scheduling of resources to use, and the type of data to exchange. The support information for tracking purposes represents the *content of a context* whether of type $\mathcal{U}$, $\mathcal{W}$, or $\mathcal{R}$. In this paper, we discuss the rationale of securing the content of these three contexts and the mechanisms that are set up for authorizing or restricting the management of this content by using specific *policies*. Management means here consultation and update.

Our security approach is built upon a forth type of context, which we extend from context and refer to as security context ($\mathcal{S}$-context). While the three aforementioned types of context are responsible for catering information on users, Web services, and resources, respectively, the security context is responsible for overseeing their $\mathcal{U}/\mathcal{S}/\mathcal{R}$-contexts (Fig. 1). Therefore the security context is specialized into three types: $\mathcal{S}$-context$_{\mathcal{U}}$ ($\mathcal{S}$ecurity context of $\mathcal{U}$ser context), $\mathcal{S}$-context$_{\mathcal{W}}$ ($\mathcal{S}$ecurity context of $\mathcal{W}$eb service context), and $\mathcal{S}$-context$_{\mathcal{R}}$ ($\mathcal{S}$ecurity context of $\mathcal{R}$esource context). Fig. 1 illustrates 2 types of link between contexts: *to extend* for specialization purposes and *to oversee* for tracking purposes.

By promoting security context, our objective is to track all the concerns and threats that affect the content of a context, to deploy appropriate measures based on previous security contexts, and to adjust the measures subject to the feedbacks obtained out of this tracking. Some of the elements that could be identified through a security context are multiple like the regular actions for identification and encryption/decryption, the types of violation that targeted the protective measures, and the corrective actions that are run for fixing misuse or alteration situations.

Section 2 discusses the rationale of a security con-

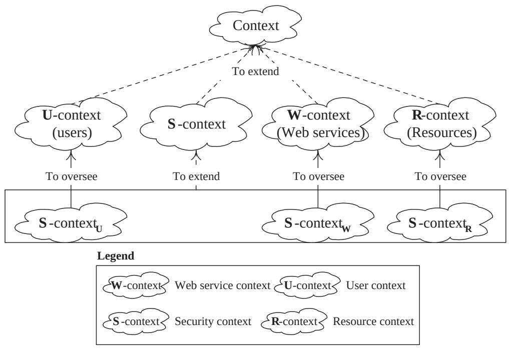

Figure 1: Types of context in a Web services environment.

text and overviews the Web services personalization project as a running case. Section 3 explains our approach to securing the management of the content of contexts by using security context. Section 4 presents the prototype and Section 5 overviews some related works. Conclusions are drawn in Section 6.

2 BACKGROUND

2.1 Why a Security Context?

A security context is a state of the working environment that requires taking one or more security actions. A security context is formed by a set of information collected from the user's environment and the application environment and that is relevant to the security infrastructure of both the user and the application (Kouadri Mostéfaoui and Brézillon, 2004).

We expect that a security context will highlight the elements that define the security of the context content of the components that are engaged in Web services personalization. These elements are the encryption algorithm, the authentication operation, the confidentiality tool, etc. For instance, if a Web service needs to be authenticated prior to managing its context, it will have to comply with this procedure. In addition we expect that the security context will limit the changes in the content of a context. For instance, a Web service could refrain from submitting consultation requests to its context because of the interception threats that originate from a resource.

2.2 Application Case: Web Services Personalization

A complete description of the Web services personalization project, which integrates the three types of context (i.e., $\mathcal{U}$-context, $\mathcal{W}$-context, and $\mathcal{R}$-context) into its operation is given in (Maamar et al., 2005). In the following we only overview this project by focussing on the interactions that occur during Web services personalization.

Fig. 2 illustrates the interactions that take place during context-based personalization of Web services before considering security. When a user selects a Web service, he continues afterwards with the personalization of this Web service according to time or location preferences. Time preference is twofold: when the execution of the Web service should start, and when the outcome of this execution should be delivered to the user back. Location preference is attached to user and is twofold: where the execution of the Web service should occur, and where the outcome of this execution should be returned to the user.

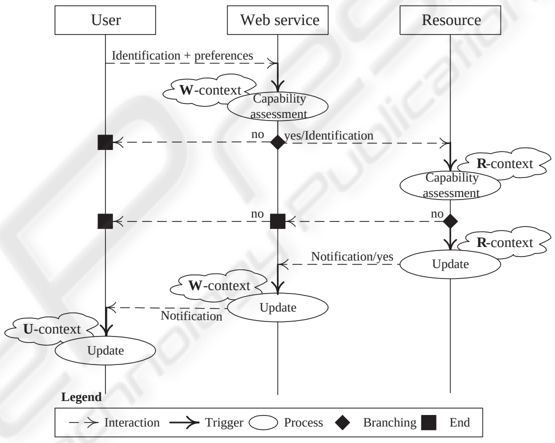

Figure 2: Interactions during Web services personalization.

Once the preferences of a user are submitted to the Web service, this latter ensures that the dates and locations are valid as conflicts might emerge during operation (e.g., delivery time occurring before execution time). Before the identification of the resources on which it will operate, the Web service checks its $\mathcal{W}$-context with regard first, to the number of current active participations in compositions *vs.* the maximum number of active participations in compositions and second, to the next period of unavailability. After a positive check of the $\mathcal{W}$-context content (to be secured), the identification of a resource can now start. A resource mainly needs to accommodate the starting time of the execution of a Web service, and the time that the execution of a Web service lasts. The outcome of this execution is tightened to the delivery time as per user's indication. To this purpose a resource checks its $\mathcal{R}$-context with regard first, to the next periods of time that will feature the execution of Web services and second, to the next period of maintenance. After a positive check of the $\mathcal{R}$-context content (to be secured), the resource notifies the Web service, which itself proceeds with notifying the user when the response of his request is ready for delivery.

The various notifications among these three components result in updating the content of the different contexts (Fig. 2). It will be shown in the next section the way these update operations are monitored.

3 POLICIES TO PROTECT CONTEXT

3.1 Protection Strategy

The aim of the security context is twofold. The first aim is to announce the security mechanisms, which guarantee the necessary protection of $\mathcal{U}$/$\mathcal{W}$/$\mathcal{R}$-contexts in terms of authentication, message safety, and data integrity. This aim was already investigated in our previous research on context ontologies (Maamar et al., 2006b). A non-authorized access to a context could result in an inaccurate assessment of multiple elements like location of user, execution status of a Web service, or capacity of a resource. The second aim, focus of this paper, is to keep track at the level of the security context ($\mathcal{S}$-context$_{\mathcal{U}/\mathcal{W}/\mathcal{R}}$) of all the operations that users/Web services/resources initiate over the content of their respective $\mathcal{U}$/$\mathcal{W}$/$\mathcal{R}$-contexts.

Operations on a context content are multiple ranging from consultation and modification to content exchange between contexts. The sensitive nature of the content of contexts, as emphasized by the first aim of the security context, has motivated the development of a set of policies that frame the performance of these operations based first, on the status of the environment surrounding users/Web services/resources and second, on previous experiences of running similar operations. For instance, a Web service could refrain from consulting its $\mathcal{W}$-context's parameters if this Web service finds out that the resource, on which it operates, can intercept and modify content. In this paper, we use Ponder (Damianou, 2002) to specify the various policies per type of context and per type of operation to be run over this context content.

Fig. 3 illustrates our proposed policy-based architecture for protecting a context content. The architecture highlights three components namely user, resource, and Web service, and a repository of policies (to be associated with an authorization engine). To keep Fig. 3 clear, the way these three components are ”plunged” in the surrounding environment is represented with the shape overlapping between ”user/resource/Web service” rectangle and ”environment” cloud. Each component binds to a context that is overseen by a security context. A component carries out operations over the content of its context upon receiving approval from the repository of policies (i.e., the authorization engine). This approval is the result of triggering policies based on the inputs that the repository receives after assessing the environment and consulting the security context. We have developed two types of policies in compliance with the types of policies in Ponder: *authorization* and *restriction*. As example of a policy specification is given in Section 3.2. The input from the environment is about user (location, status$_{\{busy,resting,etc.\}}$, etc.), Web service (status$_{\{active,suspended,etc.\}}$, provider identifier, QoS, etc.), and resource (load$_{\{high,low,etc.\}}$, provider identifier, QoS, etc.). The input from the security context is about previous experiences, which have dealt with the content of context. Previous experience refers to a similar operation over a context content that occurred in the past and was reported at the level of the security context. Any change in the content of a context is immediately communicated to the security context. A change is illustrated by setting new values to parameters of contexts, modifying the update flag of a parameter from ”permitted” to ”unpermitted”, modifying the consultation flag of a parameter from ”authorized” to ”unauthorized”, etc. In Fig. 3, number duplication means concurrent operations.

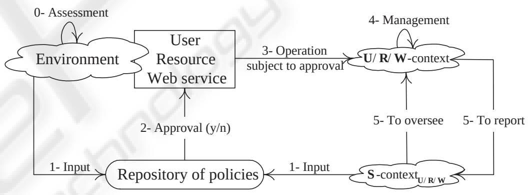

Figure 3: Architecture for content protection of context.

Our development strategy for a security context of types user/Web service/resource ($\mathcal{S}$-context$_{\mathcal{U}/\mathcal{W}/\mathcal{R}}$) consists of three steps: threat identification, security-context organization, and policy specification.

Threat identification step consists of listing the threats that could abstain a component (user/resource/Web service) from managing the content of its context. We classify the threats according to their origin.

- Threats on the content of $\mathcal{U}$-context of user primarily originate from Web services. Once a Web service is selected for participation in a composition, the Web service could turn out to be untrustworthy during run-time. Indeed, subject to personalization at the levels of execution time or execution location, a Web service may aim at modifying some preferences of user without consulting him. A motivate for the modification could be the large number of user requests that the Web service has accepted to satisfy, exceeding thus its capabilities. Since $\mathcal{U}$-context's and $\mathcal{W}$-context's parameters overlap (Maamar et al., 2005), the user is requested to modify the values of some his $\mathcal{U}$-context's parameters in accordance with the latest

Table 1: Description of $\mathcal{S}$-context$_{\mathcal{U}/\mathcal{W}/\mathcal{R}}$'s parameters and their instantiation.

Parameter & Description
Label context: identifies the context of the component that the security context is associated with.
Operation source: identifies the component that intends running an operation over a context content.
- **Operation type**: indicates if it is a consultation or modification operation.
- **Source trustworthiness**: indicates how much the component that binds to a context trusts the component that intends running an operation over this context content (null if both components are the same).
Parameter list: indicates the parameters of a context that are included in the operation of the component.
- **Last operation(s) outcome**: identifies the performance outcome in terms of success or failure of a similar operation(s) that the component binding to a context has received in the past from another component.
- **Operation validation and outcome**: indicates if the current operation over a context content is approved or denied (if *trustworthiness of source* is set to null, then *operation validation* is by default set to approved) and what the outcome of this performance is in terms of success or failure.
Date: identifies the time of updating the parameters above.
Parameter & Instantiation
Label context: $\mathcal{U}$-context$_1$ (context of user$_1$).
Operation source: Web service$_2$.
- **Operation type**: modification.
- **Source trustworthiness**: high (user$_1$ highly trusts of Web service$_2$).
Parameter list: parameter$_1$, parameter$_2$.
- **Last operation(s) outcome**: null (first time).
- **Operation validation and outcome**: approved/failure (the operation of Web service$_2$ was approved but its execution failed).
Date: 5/5/2005.

Remark: component refers to user, Web service, or resource; context refers to $\mathcal{U}/\mathcal{W}/\mathcal{R}$-context.

changes in the $\mathcal{W}$-context's parameters. If the Web service is classified as untrustworthy and a similar modification request has repetitively been issued by the same Web service, restriction policies should prevent the user from modifying $\mathcal{U}$-context. Details on the trust level of a Web service and type of request are contained in the security context (see Step 2 for details). Contrary to restriction policies, authorization policies permit the modification of the $\mathcal{U}$-context's parameters if for example the recent changes in the preferences of user at the Web service level still meet the user requirements.

- Threats on the content of $\mathcal{W}$-context of a Web service have two origins: user and resource. Regarding the user origin, the aforementioned scenario that describes the threats on $\mathcal{U}$-context of user is still valid once the threat direction is reversed. This time the user, instead of the Web service, aims at modifying his preferences without consulting the Web service. The Web service either accepts or rejects modifying the $\mathcal{W}$-context. Modification rejection could be motivated by the lack of resources on which the Web service would run. And modification acceptance could be motivated by the confidence level that the Web service aims at increasing towards this user.

  Regarding the resource origin, a Web service needs computing resources on which it operates. In addition to the overlapping situation between $\mathcal{W}$-context's and $\mathcal{R}$-context's parameters (where a change in $\mathcal{W}$-context has to be reflected on $\mathcal{R}$-context too), the resource can request some private data from the $\mathcal{W}$-context of a Web service for different purposes related to tracing current executions or scheduling forthcoming executions. A resource could accept additional Web services for execution if it could establish the various participations of this Web service in other compositions. While this threat (private-data access) could be addressed using authentication, the resource needs to be informed about the authentication mechanism that it needs to comply with prior to any attempt of consulting $\mathcal{W}$-context. This mechanism type is known as per the first aim of the security context (Maamar et al., 2006b).

- Threats on the content of $\mathcal{R}$-context of a resource originate from Web services. The aforementioned scenario that describes the threats on $\mathcal{W}$-context of

```
inst autho+ AuthorizationModification{
    subject   s = per Web service_i
    target    t = /User_j
    when      s.trustworthinessofsource(”high”) & s.operationtype(”modification”) & s.lastoperationoutcome(”success”)
    action    ModifyContext(1[t.set(parameter,value)]⋆) & Update(t.set(operationvalidation,approved))
              & Update(t.set(operationoutcome,success—failure))}
```

Figure 4: Sample of an authorization policy in Ponder.

a Web service is still valid once the threat direction is reversed. This time the Web service, instead of the resource, aims at modifying some execution parameters without consulting the resource. The resource either accepts or rejects modifying the $\mathcal{R}$-context. Modification acceptance could be motivated by the availability of the resource on which the Web service was planned running. And modification rejection could be rejected by the repetitive requests that originate from this Web service.

Security-context organization step is to identify the parameters of the security context ($\mathcal{S}$-context$_{\mathcal{U}/\mathcal{W}/\mathcal{R}}$) per component type so first, overseeing the changes in the content of contexts properly happens and second, building a bank of previous experiences occurs (Fig. 3). At this stage of our research, we have identified the following parameters independently of the type of security context (Table 1). In the second part of this table, we provide an instantiation example of these parameters.

Policy specification step consists of working on the policies that manage the content of $\mathcal{U}$/$\mathcal{W}$/$\mathcal{R}$-contexts. This is detailed in Section 3.2. Policies permit a dynamic management of the content of a context. Each management request is verified using policies. Furthermore policies can be changed on-the-fly in response to changing conditions.

3.2 Management Strategy

Our motive for adopting policies is to frame the operations that are carried out over the content of contexts according to the state of the surrounding environment and previous experiences. We selected Ponder for policy specification in compliance with some requirements that need to be satisfied like expressiveness, simplicity, and scalability (Tonti et al., 2003).

Ponder permits developing different types of policies like authorization (positive or negative) and obligation. Fig. 4 shows an authorization policy in Ponder to modify the content of a context. The modification request originates from Web service$_i$ to $\mathcal{U}$-context of user$_j$. The conditions that this request is subject to are associated with `when` label and summarized as follows: trustworthiness level of Web service$_i$ is *high*, type of operation is *modification*, and outcome of a similar operation in the past is *success*. When these conditions are satisfied, actions are performed: set the values of the parameters of $\mathcal{U}$-context, update the operation validation with approval, and finally update the operation outcome with either success or failure.

4 PROTOTYPE

We overview the implementation of securing contexts during Web services personalization. This implementation continues the prototype we developed in (Maamar et al., 2005). Fig. 5 illustrates the architecture of the prototype. Seen from a horizontal perspective, the prototype shows the different relationships that exist first, between components (i.e., user, Web service, and resource) and component contexts, and second, between component contexts and security contexts. These relationships are denoted by content management and supervision, respectively. Content management needs to be approved by an authorization engine, which is fed with the outcome of triggering policies defined à la Ponder, the current status of the environment, and previous experiences related to this content management.

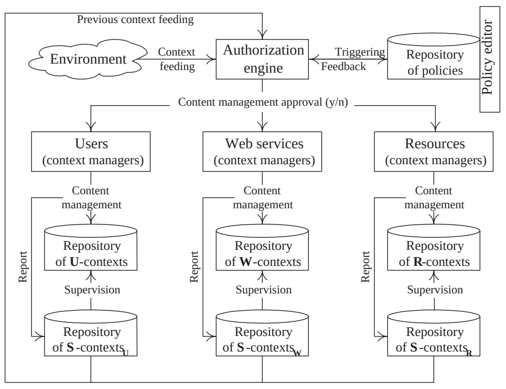

Figure 5: Architecture of the prototype.

Seen from a vertical perspective, the prototype shows the triple (component, component context, security context). The content management of any component context is reported at the level of the security context. This report is stored for later use by the authorization engine as part of previous experiences of content management.

5 RELATED WORK

In security, the use of policies is usually geared towards the specification of security mechanisms that must ensure authentication, message privacy, and authorization of Web services. We report such efforts.

In (Anderson, 2004), Anderson adopts the Web Services Policy Language (WSPL) to express policies and achieve Web services interoperability. She claims that a Web service has various aspects and features that can be controlled or described using policy rules. Examples of such aspects are authentication, quality-of-service, privacy, and reliable messaging.

Other languages for policy specification exist such as the Web Service Policy Framework (WS-Policy) (Nolan, 2004). A WS-Policy specification defines a syntax and semantics for service providers and service requestors to describe their requirements, preferences, and capabilities. The syntax provides a flexible and concise way of expressing the needs of each domain in the form of policies. A domain in this context is a generic field of interest that applies to the service and can illustrate one of the following aspects: security, privacy, application priorities, user account priorities, and traffic control.

Other service-specific policies have been proposed. Privacy policies discussed in (Yee and Korba, 2004) are an example. Yee and Korba propose a privacy policy negotiation approach to protect privacy of Web services users. Along the same direction, Indrakanti et al. make use of the XML Access Control Language (XACL) to specify authorization policies for patient records in healthcare systems implemented as Web services (Indrakanti et al., 2004).

6 CONCLUSION

In this paper we presented a policy-based approach that aims at securing the contexts associated with Web services, users, and resources. To authorize any context change, we suggested the development of a security context that reports on the strategies that protect a context using authorization and restriction policies. These policies protect context from alteration or misuse risks by framing the management operations over this context. By promoting security context, our objective was to track all the concerns and threats that affect the content of a context, to deploy appropriate measures based on previous security contexts, and to adjust the measures subject to the feedbacks obtained out of this tracking.

REFERENCES

Anderson, A. H. (2004). An Introduction to The Web Services Policy Language (WSPL). In *Proceedings of The 5th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY'2004)*, New-York, USA.

Benslimane, D., Mamaar, Z., and Ghedira, C. (2005). A View-based Approach for Tracking Composite Web Services. In *Proceedings of The Third European Conference on Web Services (ECOWS'2005)*, Vaxjo, Sweden.

Coutaz, J., Crowley, J. L., Dobson, S., and Garlan, D. (2005). Context is Key. *Communications of the ACM*, 48(3).

Damianou, N. C. (2002). *A Policy Framework for Management of Distributed Systems*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, Department of Computing.

Indrakanti, S., Varadharajan, V., and Hitchens, M. (2004). Authorization Service for Web Services and its Implementation. In *Proceedings of The IEEE Int. Conference on Web Services (ICWS'2004)*, San Diego, California, USA.

Kouadri Mostéfaoui, G. and Brézillon, P. (2004). Modeling Context-Based Security Policies with Contextual Graphs. In *Proceedings of The Workshop on Context Modeling and Reasoning (CoMoRea'2004) held in conjunction with The 2nd IEEE Int. Conference on Pervasive Computing and Communication (PerCom'2004)*, Orlando, Florida, USA.

Maamar, Z., Benslimane, D., and Narendra, N. C. (2006a). What Can Context do for Web Services? *Communications of the ACM*. (to appear).

Maamar, Z., Kouadri Mostéfaoui, S., and Mahmoud, Q. H. (2005). On Personalizing Web Services Using Context. *Int. Journal of E-Business Research, Special Issue on E-Services*, 1(3).

Maamar, Z., Narendra, N. C., and Sattanathan, S. (2006b). Towards an Ontology-based Approach for Specifying and Securing Web Services. *Journal of Information & Software Technology, Elsevier Science Publisher*. (to appear).

Nolan, P. (2004). Understand WS-Policy processing. Technical report, IBM Corporation.

Tonti, G., Bradshaw, J., Jeffers, R., Montanari, R., Suri, N., and Uszok, A. (2003). Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder. In *Proceedings of The Second Int. Semantic Web Conference (ISWC'2003)*, Sanibel Island, Florida, USA.

Yee, G. and Korba, L. (2004). Privacy Policy Compliance for Web Services. In *Proceedings of The IEEE Int. Conference on Web Services (ICWS 2004)*, San Diego, California, USA.