Context-aware Security@run.time Deployment

Wendpanga Francis Ouedraogo[1], Frederique Biennier[1], Catarina Ferreira Da Silva[2] and Parisa Ghodous[2]

[1]*Université de Lyon, INSA-Lyon, Villeurbanne Cedex, France*

[2]*Université de Lyon, Université de Lyon 1, LIRIS, CNRS, UMR 5205, 20 Avenue Albert Einstein, 69621 Villeurbanne Cedex, France*

{*wendpanga-francis.ouedraogo, frederique.biennier, catarina.ferreira-da-silva, parisa.ghodous*}*@liris.cnrs.fr*

Keywords: Context Aware Security, Execution Context, Security Patterns, Security Policy, Security as a Service.

Abstract: Taking advantage of the agility and interoperability provided by Service Oriented Architecture (SOA), Web 2.0 and XaaS (Anything as a Service) technologies, more and more collaborative Business Processes (BP) are set ”on demand” by selecting, composing and orchestrating different business services depending on the current need. This involves re-thinking the way information, services and applications are organized, deployed, shared and secured among multi-cloud environment. Fitting this de-perimeterized and evolving execution context requires organising the service protection in a dynamic way in order to provide an up to date and consistent protection. To fit this goal, we propose to integrate the different protection requirements defined according to the business environment in a single security policy. Then we plug a context-aware security deployment architecture on the cloud service middleware to analyse both the security policy and the execution context to select, compose and orchestrate the convenient protection means. A proof of concept built on Frascati middleware is used to evaluate the impact of this ”on-line” security mediation.

1 INTRODUCTION

In a highly competitive environment, enterprises are more and more involved in collaborative strategies to provide complex services and outstanding products fitting the customers requirements. Service-Oriented Architecture (SOA) and Cloud platforms provide agile and interoperable supports to compose, share and deploy on the fly complex service-based workflows. The development of such a de-perimeterized and agile Information System challenges the development of dynamic protection strategy, as threats and vulnerabilities evolve continuously depending on to both organizational and deployment platforms contexts.

The security policy model provided by the OASIS allows outsourcing security mechanisms from business services. Nevertheless, this model may lead to multiple security policies enactment as services are ”replicated” while they are composed to set a new business process. This may lead to inconsistent protection compared to the up-to-date corporate protection strategy.

To overcome this limitation, we propose to extend this security policy model to integrate context information, avoiding replicating the policy depending on the business and/or execution context. Then, this unified security policy is used as a Model@run.time by a mediation service to select, compose and orchestrate on the fly the security services deployment depending on the business context and execution environment.

After presenting a motivation example and the state of the art in section 2, we introduce our context-aware security model focusing on the business resource protection in section 3. We detail its implementation in section 4 and evaluate its performance in section 5.

2 CONTEXT AND MOTIVATION EXAMPLE

Motivations for defining a unified security policy have been found in previous projects in Collaborative Networked Organisation (CNO) and Dynamic Supply Chain environments. The reduced time-frame and the fast changing environment of these CNOs call for an opened and agile IT support. This is partly fulfilled thanks to the composition / orchestration / elastic deployment mechanisms provided by SOA, Web 2.0,

XaaS (Anything as a Service) technologies. Developing such cloud-based business service reusing ability requires to-rethink the way business service and data are protected to fit their a priori unknown execution context.

2.1 Motivation Example

This CNO dynamic protection context can be illustrated with a use case picked from a previous project where a mechanical engineering SME is involved in different CNOs. This SME uses a key business service to validate mechanical specifications of new products. This business service may be invoked in different business processes (BPs):

- The corporate *Computer Aid Design (CAD)* modelling system can invoke this service to check the intermediate specification consistency (75% of the invocations),
- The *Product Lifecycle Management (PLM)* system can invoke the service to check the product information before integrating data in its database (10% of the invocations),
- The service can also be used by some of the partners to validate the engineering requirements they send. In such case, authorized partners can invoke the service from their own *CAD* system (15% of the invocations).
  As the data produced and acceded by the validation service has a high patrimonial value for the enterprise, different protections must be deployed:
  - Strong authentication to support specification traceability, i.e., knowing who has achieved/-worked on the specifications.
  - Restricted access control to allow only people from the enterprise or some authenticated partners to accede the service.
  - Exchanged data protection with cryptographic algorithm.

These protection requirements lead to identify 3 execution contexts related to the validation service:

- *Context 1* - Internal specification checking: This service is invoked by the corporate CAD services under the control of members of the enterprise in a safe environment. In this context, no extra protection is required as the calling service can be identified.
- *Context 2* - Certified requirement checking: This service is invoked by the PLM service under the control of members of the enterprise to check engineering data validity before storing them. As it is used to implement a certification, non-repudiation mechanism is required, involving to capture user identity, check input and output information integrity.
- *Context 3* - Collaborative specification checking: This service is invoked by a partner from its CAD system to validate the specifications sent as engineering requirements. Due to business constraints, authentication, authorization and non-repudiation are necessary, whereas the open execution platform requires data encryption.

Most of the time, the business service and the protection means are orchestrated before being deployed, without checking the global consistency of the different policies on a given asset. To overcome this limitation, we propose to adapt the security policy dynamically so that the protection means fit the (may be new) business context.

2.2 State of the Art

To overcome the ”lack of trust” and ”unsuitable security policy deployment” pointed out by different surveys (Heiser and Nicolett, 2008) (Ban et al., 2010) regarding Web-based collaborative organizations, security requirements must be integrated in process models. Some annotation-based solutions have been developed to integrate security services in BPMN (Business Process Model and Notation)[1] descriptions (Rodríguez et al., 2007) (Ouedraogo et al., 2013). As these annotations are related either to a particular BPMN object or connector, extensions are required to define a global and consistent end to end process protection.

More recently, the OASIS Service Reference Architecture[2] provides a set of models to ”outsource” the security services (namely Confidentiality, Integrity, Availability, Authentication, Authorization, Non-Repudiation) from the business service. Moreover, security protocols and standards such as WS-*, XACML[3] or SAML[4] have been defined to support interoperable implementation of the necessary security mechanisms.

Different works are based on the OASIS outsourced security policy model to integrate security issues in services composition. Some use security requirements as constraints while selecting and com-

[1] http://www.omg.org/spec/BPMN/2.0/

[2] http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html

[3] http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[4] http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.html

posing services (Bartoletti et al., 2005) whereas other define various calculus algebra to decide if a right can be granted or not (Bartoletti et al., 2006). Paying attention on Business Process (BP) protection requirements, (Lang and Schreiner, 2009) has proposed a Model-Driven Security approach (MDS) to generate security policies.The generation process is achieved according to a static environment vision (perimeterized process and well-known deployment platform), leading to define different policies depending on the business context. This makes the policy management complex and limits a consistent protection evolution as modifications are achieved locally. Furthermore, while some earlier works focus mostly on access control (Wolter et al., 2009), extensions are now defined to capture protection requirements while designing BP (Lucio et al., 2014). In former works, we have proposed a simplified risk analysis knowledge base to capture protection requirements used to generate the convenient policy assertions (Ouedraogo et al., 2013). Even if this user-oriented security engineering strategy allows to identify protection requirements and mitigation means adapted to the collaboration context, it can lead to inconsistent protection as each context-dependent protection policy is defined separately. To overcome this limitation, the different protection strategies must be gathered in a single reference policy attached to each business service and deployed in a business context-aware mode to mitigate the security risks associated to contextual vulnerabilities and threats.

3 CONTEXT AWARE SECURITY SPECIFICATION

We focus on business service protection while cloud-based deployment allows reusing these protections in different business contexts. To avoid inconsistent protection specification, we extend the resource model introduced in the MDS approach to integrate context related information so that a unique security policy can be set. To maintain up-to-date protection, security patterns associated to risks mitigation best practices are also introduced.

3.1 Resource Model

A resource may be a service, some information used by the service or even a part of a process. Each resource *(Res_k)* can be characterized by a name *(ResN)*, a type *(ResT)* which can be business service/service data or a data, a resource application layer *(ResL)* (business, service, or infrastructure). A resource can include sub resources *(SubResN)* (for e.g. a business service can include set of others business services (sub services) and each sub service handles a set of data (see eq. (1)). As a consequence resource are defined in a recursive way.

$$Res_k = (ResN; ResT; ResL; \{SubRes_s\})\ 0 < s <= Ns; \quad (1)$$

Where *"Res_k"* is the resource, "k" the resource number, *"$SubRes_s$"* the name of the sub resource, "s" the sub resource number, and "Ns" the total of the sub resources.

We enrich this traditional resource definition with a "use case" part related to a particular business extension context. This allows to define the different mitigation means associated to a particular business context (see Fig. 1).

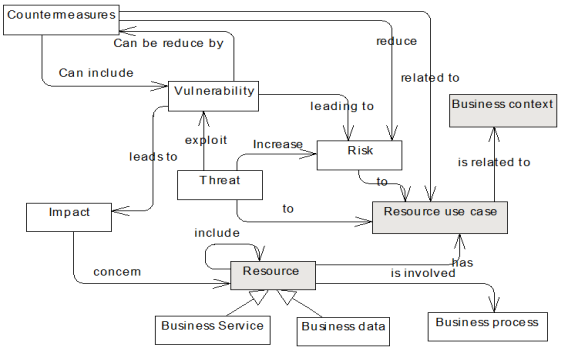

Figure 1: Security concepts associated to business service protection.

3.2 Execution Context Specification

We define the execution context *(ExCtx)* as a set of functional *(FntCtx)*, organizational *(OrgCtx)* and technical *(TecCtx)* specifications, which determine the choice of security countermeasures to perform (see eq. (2)).

$$ExCtx = (FntCtx, OrgCtx, TecCtx) \quad (2)$$

The *Functional Context (FntCtx)* describes the resource set of Functional Requirement *(FntReq)* i.e. the type of information (strategic, personal, financial...) used or handled by the resource and the resource information sensitive level (top secret, secret, confidential, restricted, unclassified) related to its importance (see eq. (3)).

$$FntCtx = \{FntReq_f\}\ 0 < f <= Nf \quad (3)$$

Where *"$FntReq_f$"* is a functional requirement, "f" requirement number, and "Nf " the total of the functional requirements. Each *FntReq* is defined as a tuple (eq. (4)):

- *FntReqId*: is the id of functional requirement,
- *FntReqT*: is the information type (strategic, personal, financial...) used or handled by the resource,
- *FntReqV*: is the sensitive level (top secret, secret, confidential, restricted, unclassified) indicating its importance based on the risk analysis method.

$$FntReq = (FntReqId, FntReqT, FntReqV) \quad (4)$$

The *Organizational Context (OrgCtx)* is mostly focused on access control (authentication and authorization), availability and non-repudiation. It allows defining the obligations and the restriction constraints in terms of access control, providing answers to the following questions: Who can access and interact with the resource? From which locality and at which time? This Organizational Context *(OrgCtx)* includes a set of organizational requirement *(OrgReq)* (see eq. (5)).

$$OrgCtx = \{OrgReq_o\} \ 0 < o <= No \quad (5)$$

Where $OrgReq_o$ is an organizational requirement, "o" the requirement number, and "No" the total of the organizational requirement. Each *OrgReq* has a tuple (see eq. (6)):

- *OrgReqId*: is the organizational requirement id,
- *OrgReqT*: is the type of organizational requirement (Who, When, FromWhere).
- *OrgReqW*: defines the way that the requirement is defined ("UserGroup" or "individually" for the "who" requirement, "date" or "dateTime" for "when", and "country" or "region" or "IPAddress" for "FromWhere").
- *OrgReqV*: is the value related to OrgReqW, i.e. the specific user or service which can invoke the resource, its localization and the period the resource can be invoked.

$$OrgReq = (OrgReqId, OrgReqT, OrgReqT, OrgReqV) \quad (6)$$

The *Technological Context (TecCtx)* includes a set of Technical Requirements *(TecReq)* related to the network, devices and deployment platform involved in the user / resource interaction at runtime (see eq. (7)).

$$TecCtx = \{TecReq_t\} \ 0 < t <= Nt \quad (7)$$

Where $TecReq_t$ is a technical requirement, "t" the requirement number, an "Nt" the total of the technical requirement.
Each $TecReq_t$ defines which technical mean (networks, devices) is used to access the resource. It is defined as a Tuple including (eq. (8)):

- *TecReqId*: is the technical requirement id.
- *TecReqT*: is the type of technical element (device, network, protocol, etc.) authorized to access or interact with the resource.
- *TecReqV*: is the value (Smartphone, PC, IP address) related to type of technical element.

$$TecReq = (TecReqId, TecReqT, TecReqV) \quad (8)$$

To support a context-aware security deployment, we integrate this context information in the resource model (see eq. (9)):

$$Res_k = (ResN; ResT; ResL; \{SubRes_s\}, \{ExCtx_e\}) \ 0 < e <= Ne \quad (9)$$

Where $ExCtx_e$ is a particular execution context, "e" is the execution context number, and "Ne" the total of the execution number.

Based on this model, listing 1 presents the *ValidateSpec* resource associated to the specification checking service from our motivation example.

Listing 1: XML-based syntax representation of the *validateSpec* resource from our motivation example.

```
1  <res resN="/MechanicalSystemSpec/validateSpec" resT="
       Service" resL="Service">
2    <exCtx id="1">
3      <fntCtx>
4        <fntReq fntReqId="2" fntReqT ="strategic"
             fntReqV ="Secret"/>
5      </fntCtx>
6      <orgCtx>
7          <orgReq orgReqId="1" orgReqT="who" orgReqW="
               UserGroup" orgReqV ="{CADService}"/>
8      </orgCtx>
9      <tecCtx>
10         <tecReq tecReqId="2" tecReqT="Network/DNS"
               tecReqV ="{mechanicalCompagny.com}"/>
11       </tecCtx>
12   </exCtx>
13   <exCtx id="2">
14     <fntCtx>
15       <fntReq fntReqId="2" fntReqT ="strategic"
             fntReqV ="Secret"/>
16     </fntCtx>
17     <orgCtx>
18       <orgReq orgReqId="1" orgReqT="who" orgReqW="
             UserGroup" orgReqV ="{PDMService}"/>
19     </orgCtx>
20     <tecCtx>
21       <tecReq tecReqId="2" tecReqT="Network/DNS"
             tecReqV ="{mechanicalCompagny.com}"/>
22     </tecCtx>
23   </exCtx>
24   <exCtx id="3">
25     <fntCtx>
26       <fntReq fntReqId="1" fntReqT ="strategic"
             fntReqV ="Secret"/>
27     </fntCtx>
28     <orgCtx>
29       <orgReq orgReqId="1" orgReqT="who" orgReqW="
             UserGroup" orgReqV ="!{CADService,␣
             CADService}"/>
30     </orgCtx>
31     <tecCtx>
32       <tecReq tecReqId="2" tecReqT="Network/DNS"
             tecReqV ="!{mechanicalCompagny.com}"/>
33     </tecCtx>
34   </exCtx>
35 </res>
```

This listing describes the *validateSpec* resource (Line 1) and its different execution contexts: Lines 2-13 for Context 1, Lines 14-25 for Context 2 and Lines

26-37 for Context 3. Access control features are defined for each context (see line 7 for Context 1, line 19 for Context 2 and line 31 for Context 3).

3.3 Context Aware Pattern Model

To generate the adapted protection means, the adapted countermeasures are selected from the OASIS security taxonomy. Our countermeasure pattern model is a tuple (eq. (10)):

$$Pat_p = (PatN, PatG, PatL, PatM, \{PatCtx_c\}, PatR, \{PatSet_s\}, PatCsq) \quad where\ 0 < (p,c,s) <= (Np,Nc,Ns) \tag{10}$$

- *Pattern Name (PatN)* is a unique name that refers to one element of our security taxonomy.
- *Pattern Goal (PatG)* defines the reason for using the pattern.
- *Pattern layer (PatL)* defines the layer of the pattern (Business, Service, Technique).
- *Pattern Metric (PatM)* is the protection level provided by pattern (Very High, High, Medium, and Lower) to fit risk sensitivity level (Top Secret, Secret, Confidential, Restricted).
- *Pattern Context ($PatCtx_c$)* defines the set of contexts in which the pattern is applicable. Each $PatCtx_c$ includes a set of optional conditions split into functional conditions *(FntCdt)*, organizational conditions (*OrgCdt*) and technical conditions *(TecCdt)* related to this context (see eq. (11)).

$$PatCtx_c = (\{FntCdt_f\}, \{OrgCdt_o\}, \{TecCdt_t\}) \quad 0 < (c,f,o,t) <= (Nc,Nf,No,Nt) \tag{11}$$

The *FntCdt* includes the information type (FntCdtT) that the resource has to handle and sensitive level value (FntCdtV) of this information for which the pattern will be useful (eq. (12)).

$$FntCdt = (FntCdtT, FntCdtV) \tag{12}$$

For example data encryption pattern can be used if the information sensitive level is secret, requiring a high protection security. The *OrgCdt* (eq. (13)) defines among 3 organizational conditions which type of conditions *(OrgCdtT)* are necessary to make the pattern applicable:

- The *"who"* condition defines the type of user authorized to access the resource.
- The *"where"* condition defines the authorized location of the user.
- The *"when"* condition defines time constraints to accede to the resource.

Condition applicability *(OrgCdtW)* specifies the control strategy. For example the XACML protocol can be used if the organizational condition is defined and the access control mode is by "User-Group".

$$OrgCdt = (OrgCdtT, OrgCdtW) \tag{13}$$

The technical condition *(TecCdt)* (eq. (14)) refers to platform-dependent protection constraints. It is used to select a pattern according to the type of the technical component that is implemented (TecCdtT) and the related value (TecCdtV).

$$TecCdt = (TecCdtT, TecCdtV) \tag{14}$$

- *Pattern related (PatR)* defines a set of related patterns. For e.g., Authentication pattern can require a set of different patterns (integrity and encryption patterns) to secure authentication token.
- *Pattern Setting (PatSet)* (eq. (15)) describes the set of parameters (Set) necessary to use a pattern. These parameters are initialized according to the execution context.

$$PatSet = \{Set_s\}\ 0 < s <= Ns \tag{15}$$

Where "s" is the parameter number and "Ns" the total number of parameters. Each parameter *(Set_k)* is defined as a tuple (eq. (16)):

$$Set = (Setkey, SetValue) \tag{16}$$

Where *Setkey* is the parameter name and *SetValue* the value of the parameter.

- *Pattern Consequence (PatCsq)* describes the results or actions implemented by the pattern. The set of patterns is defined by (eq. (17)):

$$Pats = \{Pat_j\}\ where\ 0 < j <= N_j \tag{17}$$

Where "j" is the pattern number and "Nj" the total number of patterns.
Based on the protection requirement to fulfill and on information on the current deployment context, the convenient mitigation pattern can be selected to generate a security policy rule.

3.4 Reference Security Policy Model

Our formal security policy model, defined as a tuple (eq. (18)), extends the OASIS policy model. For each security criteria defined in the OASIS model, a set of policy rules is defined and related to policy context. By this way, the security mechanisms that must be deployed to provide a consistent protection can be tuned depending on the execution context.

$$Pol_x = (PolR, PolT, PolG, PolL, PolRls)\ 0 < x <= Nx \tag{18}$$

Where

- *Policy Resource (PolR)*: is the resource (Business service, data, etc.) involved in the policy.
- *Policy type (PolT)*: refers to the security service taxonomy (Authentication, Confidentiality, Integrity, Non-repudiation, Availability, etc.)
- *Policy Goal (PolG)*: defines the policy aims.
- *Policy Layer (PolL)*: is the layer (Business, Service or Infrastructure) of the involved resource for this policy.
- *Policy Rules (PolRls)*: is a set of policy rules (eq. (19)). Each policy rule *($PolRl_r$)* defines the security mechanism to apply according to the resource execution context.

$$PolRls = \{PolRl_r\}\ 0 < r <= N_r \qquad (19)$$

Where $PolRl_r$ is a policy rule, "r" the policy rule number, and "Nr" the total of policy rules. Each policy rule is also defined as a tuple (eq. (20)):

$$PolRl = (RlId, RlP, RlM, RlCtx, \{RlSet\}) \qquad (20)$$

Where:

- *Policy Rule Id (Rlid)*: is the policy rule id,
- *Policy Rule Pattern (RlP)*: is the name of the pattern which matches the resource execution context.
- *Policy Rule Metric (RlM)*: is the pattern associated metric which matches the resource execution context. It is the protection level provided by the pattern.
- *RlCxt*: includes the part of the pattern context which matches the resource execution context.
- *RlSet*: is the initialized pattern settings allowing to invoke the security mechanism.

Then all the policies associated to a resource are gathered in a single set (eq. (21)):

$$Pols = \{PolR_x\}\ where\ 0 < x <= N_x; \qquad (21)$$

Where "x" is the policy number and "Nx" the total number of policy (for all resources).

Lastly, policy attached to any resource Rk can be defined by selecting (σ) the policy in the Pols set where the policy resource (PolR) matches with resource "R_k":

$$Pols(R_k) = \sigma_{(Pols.PolR=R_k)}(Pols) \qquad (22)$$

This security policy is used to generate a security policy file attached to the resource description (WSDL[5] or WADL[6] file) that contains all the security assertions and their application context. Listing 2 shows the security policy of the specification validation operation (*validateSpec*) of the Mechanical System specification business service, namely authentication (Lines 2-14) if the calling service is not the corporate *CADService*, authorization (Lines 27-39), non repudiation (Line 15-26) and communication channel protection (Lines 40-54).

Listing 2: validateSpec service security policy expressed using XML syntax.

```
1  <pols>
2  <pol polId="1" polR="/mechanicalSystemSpec/
       validateSpec" polT="Authentication">
3    <polR rlId=1 RlP="LoginPWD", rlM="0.5">
4      <rlCtx>
5          <orgCdt orgReqT="who" orgReqW="UserGroup"
               orgReqV="!{CADService}"/>
6          <tecCdt tecReqT="Network" tecReqV="Network/
               DNS" tecReqV="!{mechanicalCompagny.com}
               "/>
7        </rlCtx>
8        <rSets> <rSet setkey="userRegistry" setValue="
             data/UserRegistry.xml"/>
9        </rSets>
10     </polR>
11   </pol>
12   <pol polId="2" polR="/mechanicalSystemSpec/
       validateSpec" polT="NonRepudiation">
13     <polR rlId="1" RlP="Log", rlM="0.5">
14       <rlCtx>
15         <orgCdt orgReqT="who" orgReqW="UserGroup"
               orgReqV="!{CADService}"/>
16         <tecCdt tecReqT="Network" tecReqV ="IP"/>
17       </rlCtx>
18       <rSets>
19         <rSet setkey="logFile" value="log.log"/>
20       </rSets>
21     </polR>
22   </pol>
23   <pol polId="3" polR="/mechanicalSystemSpec/
       validateSpec" polT="Authorization">
24     <polR rlId="1" RlP="ACL", rlM="0.5">
25       <rlCtx>
26         <orgCdt orgReqT="who" orgReqW="UserGroup"
               orgReqV="!{CADService,PDMService}" />
27         <tecCdt tecReqT="Network" tecReqV ="Network/
               DNS" tecReqV="!{mechanicalCompagny.com}"
               />
28       </rlCtx>
29       <rSets>
30         <rSet setkey="ACLFile" value="acl/
             AccessControlList.xml"/>
31       </rSets>
32     </polR>
33   </pol>
34   <pol polId="4" polR="/mechanicalSystemSpec/
       validateSpec" polT="Confidentiality">
35     <polR rlId="1" RlP="Encryption_AES_128", rlM="
           0.75">
36       ....
37       ....
38     </polR>
39   </pol>
40 <pols>
```

4 CONTEXT AWARE SECURITY DEPLOYMENT

Taking advantage of the loosely coupled strategy, we use the middleware online interception capability to intercept each service/middleware interaction and routes this interaction to our context aware architecture (Fig. 2) built as an add-on component. It consists in:

[5]http://www.w3.org/TR/wsdl

[6]http://www.w3.org/Submission/wadl/

- a *Context Aware Deployment* composite which is the core component to achieve the dynamic security deployment. This component is split into three sub components:
  - The *Context Aware Security* component analyses the calling service requests intercepted by the middleware and encapsulated into a Request object. It invokes the policy manager and the context manager to get the security policy rules associated to the business services fitting the context before invoking the required security services.
  - The *Context Manager* analyses security policies associated to the services and identifies the different policies to be applied according to the current context.
  - The *Policy Manager* identifies the list of policies fitting the context from the global policy file associated to the resource.
- The *Security as a Service* (SecaaS) composite gathers implementation of various security services (authentication, authorization, integrity controls, etc.) such as:
  - The *SecaaS* component is the composite entry point. It analyses the policies sent by the Context Manager component to identify the security services (authentication, authorization, etc.) to call.
  - The security components (*Authentication*, *Authorization*, *Integrity*, *Encryption*, *Non-Repudiation Availability*) invoke the security service depending on the pattern and parameters specified in the policy.

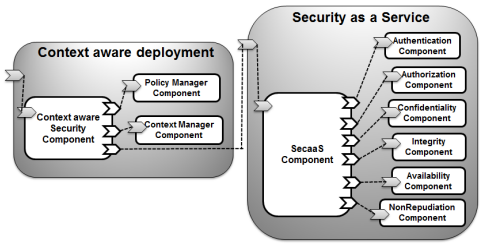

Figure 2: Context aware security architecture.

5 EVALUATION

To evaluate the impact of our Context Aware Deployment, we implement a Proof of Concept plugged on the Frascati middleware (Merle et al., 2011) extending our Model Driven Policy generator presented in (Ouedraogo et al., 2013). We then use this prototype to support various experiments on BP protection policies specification depending on shared collaborative context.

5.1 Performance Evaluation

We assume that the cloud oriented service middleware provides a unified environment to invoke business services dispatched on a multi-cloud platform. As a consequence, we set a test on a single machine environment using Frascati version 1.6 with Oracle Java Virtual Machine 1.7.0 51 on Microsoft Windows 7 Professional (32 bit) using a 2,54GHz processor Intel(R) Core(TM) 2 Duo CPU with 4Go of memory to compare the dynamic security deployment execution time with the business service and the protection service execution times.

In our example, the *"validateSpec"* of the *"mechanicalSystemSpec"* service and the related information must be protected in the different contexts. The confidentiality requirement impacts both application layer, which is in charge of the access control, traceability, and transport layer. The systematic composition of the authorization, traceability and confidentiality services may be costly (see the different execution times reported in Table 1).

Table 1: *ValidateSpec* Service execution time according to the three contexts.

Context	Security services involved mechanical Specification	Execution time (ms)
Context 1	No security mechanism is required	64
Context 2	Authentication and Non Repudiation	80
Context 3	Context 2 security services + Authorization and Confidentiality	84

The contextualized security service orchestration allows invoking only the necessary security mechanisms according to the context, avoiding both the costly over protection and the risky under protection. This dynamic protection does not impact the daily used BP and simplifies the security policy consistency controls as new protection requirements are introduced once in the service policy specification and implemented for the different collaborative BP without impacting the business process deployment.

Other performance simulations have first been conducted in order to evaluate the performance overhead involved by the context analysis at run time. We extend this benchmark to different services (see Table 2) to compare the cost of our context-aware architecture with the business service execution time. We set two reference execution contexts, one requiring no security deployment (Context 1) whereas the other requires the maximum protection (i.e systematic protection required in Context 3 from our motivation example). Total execution times are measured

Table 2: Execution times for a panel of 1000 invocations of business services where the ”no protection” rate evolves from 0% to 100%.

	Systematic protection	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
S1 (10ms)	32000	38000	35800	33600	31400	29200	27000	24800	22600	20400	18200	16000
S5 (50ms)	72000	78000	75800	73600	71400	69200	67000	64800	62600	60400	58200	56000
S10(100ms)	122000	128000	125800	123600	121400	119200	117000	114800	112600	110400	108200	106000

for 1000 invocations split among different occurrence rate for context 1 (see Table 2). The ratio comparing the context-aware secured business service execution time with the systematically secured business service execution time presents a maximum of 4,92% overhead for the bigger service to 18,75% for the smallest one when these services invocation requires the highest protection (0% of occurrence of the ”no protection context”). On the opposite, our context aware security deployment can save from 50% up to 86,89% of execution time when all invocation do not need any protection. These results show that the overhead involved by our architecture can be rather neglected (from 4,92% to 18,75% of the service execution time overhead when the highest protection is always required) compared to the large overhead introduced by the systematic invocation of (often) useless security services provided that the ”no protection” invocation context rate is greater than 30% as shown in Fig. 3.

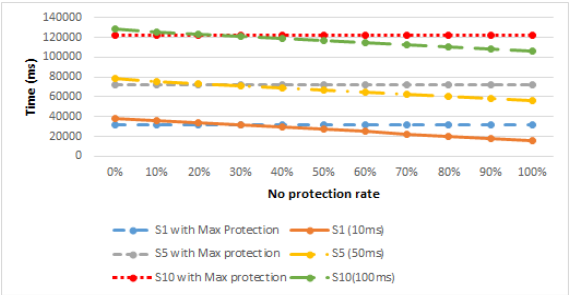

Figure 3: Variation of context aware security execution cost for three business services.

6 CONCLUSION

To secure business services used in collaborative environment, enterprises have to adapt the protection according to the execution context. To this end, we propose a context aware security model and architecture used to select and orchestrate security services at runtime. This architecture, tested on the Frascati middleware, shows that the dynamic security mediation has a rather low impact on the performance level compared with a systematic deployment of costly overprotection.

Further works will focus on the integration of more detailed platform models and on vulnerabilities monitoring loops so that our coarse-grained vision of the execution context will be refined to increase the protection efficiency.

REFERENCES

Ban, L. B., Cocchiara, R., Lovejoy, K., Telford, R., and Ernest, M. (2010). The evolving role of it managers and cios.

Bartoletti, M., Degano, P., and Ferrari, G. (2005). Enforcing secure service composition. In *Computer Security Foundations, 2005. CSFW-18 2005. 18th IEEE Workshop*, pages 211–223.

Bartoletti, M., Degano, P., and Ferrari, G. (2006). Security issues in service composition. In Gorrieri, R. and Wehrheim, H., editors, *Formal Methods for Open Object-Based Distributed Systems*, volume 4037 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg.

Heiser, J. and Nicolett, M. (2008). ssessing the security risks of Cloud Computing. *Technical report, Gartner*.

Lang, U. and Schreiner, R. (2009). Model Driven Security Management: Making Security Management Manageable in Complex Distributed Systems. In *Workshop on Modeling Security (MODSEC08) - International Conference on Model Driven Engineering Languages and Systems (MODELS)*.

Lucio, L., Zhang, Q., Nguyen, P. H., Amrani, M., Klein, J., Vangheluwe, H., and Traon, Y. L. (2014). Chapter 3 - Advances in Model-Driven Security. In Memon, A., editor, *Advances in Computers*, volume 93, pages 103 – 152. Elsevier.

Merle, P., Rouvoy, R., and Seinturier, L. (2011). A Reflective Platform for Highly Adaptive Multi-Cloud Systems. In *International Workshop on Adaptive and Reflective Middleware (ARM'11) - 12th ACM/IFIP/USENIX International Middleware Conference*, pages 14–21. ACM.

Ouedraogo, W. F., Biennier, F., and Ghodous, P. (2013). Model driven security in multi-context. In *International Journal of Electronic Business Management*, volume 11 No. 3, pages 178–190.

Rodríguez, A., Fernández-Medina, E., and Piattini, M. (2007). A BPMN Extension for the Modeling of Security Requirements in Business Processes. *IEICE - Trans. Inf. Syst.*, E90-D(4):745–752.

Wolter, C., Menzel, M., Schaad, A., Miseldine, P., and Meinel, C. (2009). Model-driven business process security requirement specification. *Journal of Systems Architecture (JSA)*, pages 211–223.