Web Services Enterprise Security Architecture: A Case Study

Carlos Gutiérrez
STL
Xaudaró, 15
28034, Madrid. (SPAIN)
34 913 48 92 61
Carlos.Gutierrez@stl.es

Eduardo Fernández-Medina
Alarcos Research Group,
Universidad de Castilla-La Mancha
Paseo de la Universidad 4
13071, Ciudad Real. (SPAIN)
34 926 29 53 00
Eduardo.FdezMedina@uclm.es

Mario Piattini
Alarcos Research Group,
Universidad de Castilla-La Mancha
Paseo de la Universidad 4
13071, Ciudad Real. (SPAIN)
34 926 29 53 00
Mario.Piattini@uclm.es

ABSTRACT
Web Services (WS hereafter) Security is a crucial aspect for technologies based on this paradigm to be completely adopted by the industry. As a consequence, a lot of initiatives have arisen during the last years setting as their main purpose the standardization of the security factors related to this paradigm. In fact, over the past years, the most important consortiums of Internet, like IETF, W3C or OASIS, are producing a huge number of WS-based security standards. Despite of this growing, there's not exist yet a process that guides developers in the critical task of integrating security within all the stages of the development's life cycle of WS-based software. Such a process should facilitate developers in the activities of web service-specific security requirements specification, web services-based security architecture design and web services security standards selection, integration and deployment. In this article we briefly present the PWSSec (Process for Web Services Security) process that is composed of three stages, WSSecReq (Web Services Security Requirements), WSSecArch (Web Services Security Architecture) and WSSecTech (Web Services Security Technologies) that accomplishes the mentioned activities, respectively. We also provide a thorough explanation of the WSSecArch (Web Services Security Stage) stage intended to design the web services-based security architecture. In addition, a real case study where this stage in being applied is also included.

Categories and Subject Descriptors
D.2.1 [**Software Engineering**]: Requirements/Specifications - *Elicitation methods (e.g., rapid prototyping, interviews, JAD)*, Software Architectures – *Domain-specific architectures.*

General Terms: Security.

SWS'05, November 11, 2005, Fairfax, Virginia, USA.

Keywords
Security, web services, software development process, software architecture.

1. INTRODUCTION
Web Services (WS hereafter) technologies have become the 'de facto' solution for Enterprise Application Integration since it enables complex business workflow integration scenarios and provides the so-demanded and so-called hyper-connectivity inter- and intra-enterprises [21]. IDC estimates that $2.3 billion was spent worldwide on total WS software in 2004, more than double the amount from the previous year. IDC expects spending to continue to increase dramatically over the next 5 years, reaching approximately $14.9 billion by 2009 [14]. Due to this fact, an enormous quantity of WS-based standards is being produced. This diversity, also found in the context of WS security [11] has made us to consider its application, from a global perspective, as a very complex and hard process to understand with a very difficult learning curve.

At present, there is still a lack of a global approach that offers a methodical development for constructing security architectures for WS-based systems. Thus, the main objective of this paper is to present the process PWSSec (Process for Web Services Security) [12]. PWSSec has been created to facilitate and orientate the development of security for WS-based systems in a way that in each one of the traditional stages for the development of this sort of systems [6], a complementary stage comprising security can be integrated. Therefore, this process can be used once the functional architecture of the system has been built or during the stages used to elaborate this architecture. In both cases, the result will be a WS-based security architecture formed by a set of coordinated security mechanisms that use the WS security standards to fulfil the WS-based system security requirements.

The main contribution of this paper is the presentation and application of the security reference architecture specified in the WSSecArch stage. A preliminary version of the WSSecArch stage was presented in [13]. In addition, a real case study that demonstrates how the WSSecArch stage of the PWSSec process can be applied in order to provide 'quality of protection' to a request/reply interaction between a WS consumer agent and a WS provider agent is developed. The allocation of the security requirements into a security web services-based architecture is

explained and the necessary security policies to be defined are stated. In addition the web services-based security reference architecture will be developed explaining how its main elements interact in order to address both functional and security requirements.

The case study presented in this article is a real development that is being carried out between three bank organizations and a state-owned company dedicated to sport and lottery gambling (hereafter SportGamblingOrg). When a participant wins a prize higher than 600 euros, he has to go to one of the predetermined bank organizations, identify himself and request the payment of his prize. The branch of the bank organization where the participant goes to get his prize has to connect with a Legacy Backend system, managed by the SportGamblingOrg organization, by means of a web service consumer agent. This web service consumer agent carries out a request/reply interaction with a web service provider agent located at the SportGamblingOrg organization. The SportGamblingOrg's web service interacts with a Central Legacy Backend system by offering three main operations: prize's payment request, prize's payment state request and bank organization's prize's payment report request. The Central Legacy Backend system is the ultimate responsible for deciding whether certain lottery ticket has a prize or not. The bank organization's branch can pay the prize to the participant if, it has previously obtained a prize's payment's confirmation from the Central Legacy Backend system.

When the Central Legacy Backend system verifies that the lottery ticket has a prize assigned that has not been paid yet, it sends a confirmation response to the SportGamblingOrg's web service provider agent that, in turn, sends the response to the bank organization's web services consumer agent.

Once the Central Legacy Backend system replies with a payment confirmation it changes the state of the lottery ticket to 'payed' state. When the bank organization's branch receives the confirmation stating that the lottery ticket given by the participant has a prize, it can proceed to carry out the real payment on behalf of the participant. Figure 1 shows the use cases that the aforementioned system should implement.

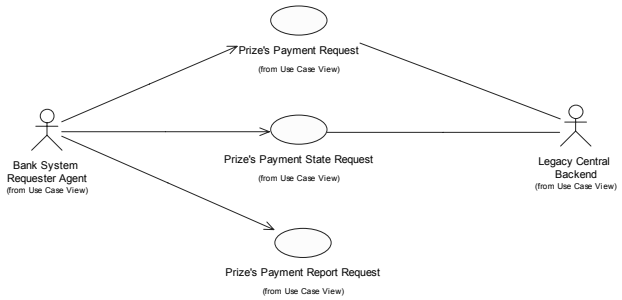

Figure 1. Use case view of the case study.

In section 2, a brief introduction to the PWSSec process is presented; in section 3, the security requirements elicited for the case study are listed; in section 4, the WSSecArch is presented in a detailed fashion and the case study is developed; in section 5, conclusions and future work are indicated.

2. PWSSec overview

In this section we provide an overall view of the PWSSec process, including its main objectives, basic principles and the stages into which it is divided.

In general terms, the main characteristics of this process are:

- Iterative and incremental. The model chosen for the PWSSec process is iterative and incremental [4, 17, 18], thereby facilitating the gradual integration of WS-based security.
- It facilitates the traceability and re-usability of the process as well as the interoperability and re-usability of the product. Both principles are derived from the practical nature it has been felt necessary to confer on the process. Traceability means the capacity for tracing the properties of the system along with the different levels of abstraction, offering controlled support for modifying and extending the system [4]. Process re-usability will allow its application in different domains, within the context of WS-based systems, while product re-usability will guarantee us the fastest possible development cycles based on proven solutions. The interoperability of the product consists in identifying the responsibility in terms of logical security services.

It includes concepts and techniques developed within the scope of Security Requirement Engineering and Risk Management and Analysis [1, 9, 10, 19, 30].

Figure 2 illustrates the stages into which the PWSSec is structured.

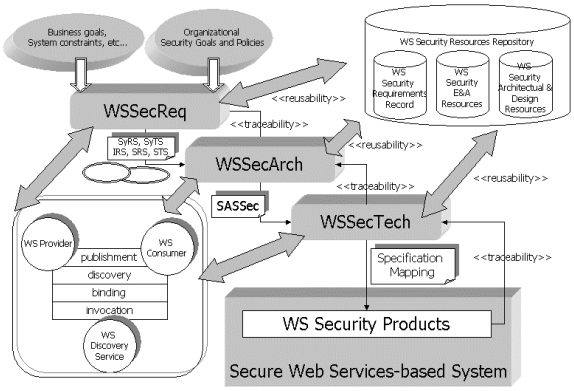

Figure 2. Stages and products in the PWSSec development process.

Each of the stages defined in PWSSec describes its inputs, outputs, activities, actors and, in some cases, there are guides, tools and techniques which complement, improve and facilitate the set of activities developed within these stages. Following, a brief description of these stages will be presented (more details could be found in [12]):

- WSSecReq (Web Services Security Requirements): The main purpose of this stage is to produce a specification (or a part of it) of the security requirements of the WS-based "to-be-constructed" system. Its input is composed by a

Table 1. Security Requirements related to message authentication, integrity, confidentiality and authorization.

Id	**001013**	The WS provider agent WS-PrizePaymentTeller shall verify the authenticity of the **request** message(s) sent by the WS consumer agent WS-BankOrg-XXX at both, HTTP transport and SOAP message-level with the aim of avoiding sophisticated attacks during the execution of use cases 'Superior Prize Payment Request', 'Superior Prize Payment Status Request', 'Superior Prize Payment Summary Report Request' in the 99.99% of the use cases' instantiation.
Quality Factor	Security	
Quality Subfactor	Data Origin Authentication	
Security Use Case	SUC-00201	
Priority	HIGH	
Criticality	HIGH	
Viability	OK	
Risk	HIGH	
Source	SportGambling	
Includes	1019	
Excludes	-	
Id	**001014**	The WS provider agent WS-PrizePaymentTeller shall require WS consumer agent WS-BankOrg-XXX to posses the necessary credentials, being pre-assigned by SportGambling organization, to be authorized to execute use cases 'Superior Prize Payment Request', 'Superior Prize Payment Status Request', 'Superior Prize Payment Summary Report Request' a minimum of 99.99% of the use cases' instantiations.
Quality Factor	Security	
Quality Subfactor	Authorization	
Security Use Case	SUC-00203	
Priority	HIGH	
Criticality	HIGH	
Viability	OK	
Risk	HIGH	
Source	SportGambling	
Includes	-	
Excludes	-	
Id	**001018**	The WS consumer agent WS- BankOrg-XXX shall protect requests, at both HTTP transport-level and SOAP message-level, transmitted so that the **winner personal info** would be only visible to the WS provider agent WS-PrizePaymentTeller resisting sophisticated attacks during the execution of use case 'Superior Prize Payment Request' in 99.99% of the use case's instantiation.
Quality Factor	Security	
Quality Subfactor	Communications Confidentiality	
Security Use Case	SUC-00204	
Priority	HIGH	
Criticality	HIGH	
Viability	OK	
Risk	HIGH	
Source	SportGambling	
Includes	-	
Excludes	-	
Id	**001019**	The WS consumer agent BankOrg-XXX shall protect the requests it transmits, at both transport- and message-level, from possible modifications, deletions and insertions over its payload due to sophisticated attacks on integrity during the execution of the use cases 'Superior Prize Payment Request', 'Superior Prize Payment Status Request', 'Superior Prize Payment Summary Report Request' a minimum of 99.99% of the use cases' instantiations.
Quality Factor	Security	
Quality Subfactor	Communications Integrity	
Security Use Case	SUC-00206	
Priority	HIGH	
Criticality	HIGH	
Viability	OK	
Risk	HIGH	
Source	SportGambling	
Includes	001013	
Excludes	-	

specification of the scope that we want to comprise during the current iteration (e.g.: if we have a definition of the Use Cases available, we can select those that we want to cover and use them as an input for the iteration), the business and security goals defined for the system as well as the part of the organizational security policy that we estimate that may impact on the system design. The output is basically formed by: i) A threat attack tree [25] associated with the WS business and application pattern [6] identified within the analyzed functionality; ii) Every built attack tree's leaf will show a threat [33] that can refined by a set of attack scenarios, defined as misuse cases according to [2, 26], organized into attack profiles [20], and represented according to the Quality if Service UML profile [24]; ii) every misuse case must have related a set of security use cases, according to Donald G. Firesmith [8], that state how the system should respond to the associated misuse case; iii) A formal specification of the security requirements for the scope of the system based on SIREN [29]. These requirements will have been derived after instantiating the WS security requirements templates associated with every security use case. This stage is supported by two repositories: i) *WS*

Security E&A Resources, that contains all the artefacts mentioned above but the security requirements specification; ii) *WS Security Requirements Record* that contains a set of generic security requirements that can be applied to WS-based systems within diverse domains [29].

- WSSecArch (Web Services Security Architecture): This stage has as its main objective to allocate and integrate the security requirements specified in the WSSecReq stage by identifiying the appropriate security WS architectural patterns and the security services derived from them. The input of this stage is composed of: i) business goals of the current iteration; ii) organizational security goals and policies taken into account during the current iteration; iii) the set of attack and security scenarios developed in WSSecReq; and iv) the set of security requirements defined in the specifications SyRS (System Requirements Specification), SRS (Software Requirements Specification), SyTS (System Tests Specification), STS (Software Tests Specification), IRS (Interfaz Requirements Specification) developed in WSSecReq stage. The output is a complete specification of the developed security architecture, called Software Security Architecture Specification (SASSec), indicating: i) how the functional requirements used as input to the stage are integrated into the specifications mentioned above; ii) what security requirements are achieved and how are the allocated in the architecture [27]; and iii) what are the security WS that need to be introduced as security mechanisms.

- WSSecTech (Web Services Security Technologies): The main purpose of this stage is to define a set of standards that will implement the Abstract Security Services identified in the previous stage. Its principal input will be the SASSec elaborated then. Output will be a description of the set of standards identified for each Abstract Security Service together with the reasoning framework that made us select it and a security architecture design. The activities carried out in this stage are the following: i) WS-based Security Standards Identification; and ii) Deployment Security Policies Definition.

3. BACKGROUND

In order to show how the WSSecArch stage has been applied to our case study, we firstly needed to apply the WSSecReq stage. As a result of the application (as explained in [12]) of this stage to the case study we obtained a set of quality-of-protection and authorization security requirements presented in Table 1.

This set of security requirements, jointly with the set of functional, misuse and security use cases, form the main input for the WSSecArch stage

4. WSSecArch

A preliminary vision of the WSSecArch was presented in [13]. In this paper we will present a more detailed view of this stage and, in particular, the application of the security reference architecture to the aforementioned case study.

In Figure 3, the steps which the WSSecArch is divided into are presented. In this paper we will focus on the Security Pattern

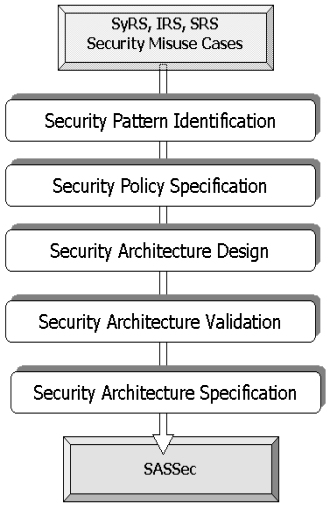

Figure 3. Tasks defined within the WSSecArch stage.

Identification, Security Policy Specification and the Security Architecture Design activities.

The WS-based security reference architecture has as its main objective to guide system designers in the task of allocating the security requirements into the security architecture and to provide an organizational and administrative basis on which functional WS, security WS and security policies can be developed, deployed and reused within the enterprise.

A description of the main elements that make up the WS-based security reference architecture can be found in [13]. The central element of this reference architecture is the Web Services Security Kernel (WSSecKern). The set of functional WS (e.g.: *WS-PrizePaymentTeller*) deployed within an enterprise reside in Security Zones. Every Security Zone has one or more WSSecKern. A WSSecKern is responsible for managing a set of WS-based security services that, in turn, implement a set of WS-based security standards. One of the WSSecKern deployed within a Security Zone will act as the master for that Security Zone. The Master WSSecKern of a Security Zone intercepts all incoming/outgoing messages directed to/from the functional WS located within the Security Zone it belongs to. In addition, the Master WSSecKern enforces the suitable security policies to all incoming/outgoing messages. The functional WS are located in Security Zones. The criteria of allocating functional WS into one Security Zone or another depends on the enterprise and project's context. For instance, the enterprise where the case study is being developed distinguishes one Security Zone: a Critical Security Zone, where all the WS provider agents that need to interact with the Central Legacy Backend system for fulfilling their tasks are deployed. Every Security Zone has assigned a Security Zone Administrator responsible for the monitoring and administration of the security and functional WS deployed within that zone.

```
            ...

<wsdl:message name="QoPProtectionRequest">
   <wsdl:part name="SOAPMsg"
          element="soapEnv:Envelope"/>
   <wsdl:part name="QoPProtectionProperties"
          element="tns:QoPProtectionProperties"/>
 </wsdl:message>

 <wsdl:message name="QoPProtectionResponse">
   <wsdl:part name="QoPProtectionResponse"
element="soapEnv:Envelope"/>
 </wsdl:message>

                                    …
 <wsdl:portType name="QoPSecurityService">
   <wsdl:operation name="protect">
     <wsdl:input message="tns:QoPProtectionRequest"/>
     <wsdl:output message="tns:QoPProtectionResponse"/>
   </wsdl:operation>
</wsdl:portType>
            ...
```

Figure 4. WSDL fragment that contains the message types and port type defined for the QoP security service.

Each WSSecKern is responsible for managing one or more security WS. Every security WS addresses a type of security requirement by means of one or more mechanisms (e.g.: authorization based on RBAC or ACL's or Integrity based on either symmetric cryptography and Message Authentication Codes or asymmetric cryptography and digital signatures), standardized by one or more WS-based security standards.

4.1 Security Pattern Identification

The mechanisms defined in the security architectural patterns could be abstracted from one or more standards or industry adopted 'de facto' solutions so that a reference security architecture with all its variants (specific solutions which the security pattern was abstracted from) can be described for certain security subfactors (e.g.: E. B. Fernandez [7] presents two WS-based security patterns that abstract XML-based firewall and assertion management solutions).

We've defined the QoP Security Pattern, abstracted from the mechanisms defined in the WS-Security specification [22], which address 'QoP' security requirements. That is, message confidentiality, message integrity and message authentication. This 'QoP' WS defines an interface for protecting and verifying the protection of SOAP messages. Figure 4 shows a fragment of the messages and port type definition of this security service. The WS-Security standard is the 'de facto' WS security standard that implements this security service.

In our case study we applied the WS QoP (Quality of Protection) architectural pattern in order to address security requirements **001013, 001018** and **001019.** That is, message authentication, integrity and confidentiality security requirements.

The WS QoP architectural pattern defines a security WS capable of protecting, and verifying the protection of, SOAP outbound/inbound messages, respectively. It defines a security policy template that should define the type of security requirement

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurityRequirements product="P456">
    <SecurityRequirement>
        <id>001013</id>
        <subfactor>data_origin_authentication</subfactor>
        <securityusecase><id>00201</id></securityusecase>
        <priority>HIGH</priority>
        <criticality>HIGH</criticality >
        <risk>HIGH</risk>
        <source><id>SportGambling</id></source>
        <includes>
                <SecurityRequirementReference><Id>001019</Id></Se
        curityRequirementReference>
        </includes>
        <excludes></excludes>
    </SecurityRequirement>
    <SecurityRequirement>
        <id>001014</id>
        <subfactor>authorization</subfactor>
        <securityusecase><id>00203</id></securityusecase>
        <priority>HIGH</priority>
        <criticality>HIGH</criticality >
        <risk>HIGH</risk>
        <source><id>SportGambling</id></source>
        <includes></includes>
        <excludes></excludes>
    </SecurityRequirement>
    <SecurityRequirement>
        <id>001018</id>
        <subfactor>communications_confidentiality</subfactor>
        <securityusecase><id>00204</id></securityusecase>
        <priority>HIGH</priority>
        <criticality>HIGH</criticality >
        <risk>HIGH</risk>
        <source><id>SportGambling</id></source>
        <includes></includes>
        <excludes></excludes>
    </SecurityRequirement>
    <SecurityRequirement>
        <id>001019</id>
        <subfactor>communications_integrity</subfactor>
        <securityusecase><id>00204</id></securityusecase>
      <priority>HIGH</priority>
                <criticality>HIGH</criticality >
                <risk>HIGH</risk>
                <source><id>SportGambling</id></source>
                <includes>
                 <SecurityRequirementReference><Id>001013</Id></S
                ecurityRequirementReference>
                </includes>
                <excludes></excludes>
        </SecurityRequirement>
</SecurityRequirements>
```

Figure 5. Security policy describing the set of security requirements of information system 456.

its instance covers (message authentication & integrity, confidentiality), the specific security mechanisms to be used and the WS specification that it deploys.

As a result of this activity, the set of security WS required and derived from WS-based security architectural patterns should have been identified. Following, the security policies related to both, functional and security WS shall be stated.

```
…
<wsdl:portType    name="PrizePaymentInterface">
        <wsdl:operation name="PrizePaymentOperation"
        wsa:Action="http://.../ws/PrizePaymentTeller/1.0/requests/PrizePaymentRequest">

          <wsdl:input
                message="tns:PrizePaymentRequestMessage"
                wsp:PolicyURIs="http://.../policies#PaymentPrizeInputMessagePolicy"
                />
                                                    …
  </wsdl:operation>
  <wsdl:operation name="PrizePaymentStatusOperation">
        <wsdl:input
                message="tns:PrizePaymentStatusRequestMessage"
                wsp:PolicyURIs="http://.../policies#PaymentPrizeStatusInputMessagePolicy"
                wsa:Action="http://.../ ws/PrizePaymentTeller/1.0/requests/PrizePaymentStatusRequest "/>
                                                    …
  </wsdl:operation>
  <wsdl:operation name="PrizePaymentReportOperation">
        <wsdl:input
                message="tns:PrizePaymentReportRequestMessage"
                wsp:PolicyURIs="http://.../policies#PaymentPrizeReportInputMessagePolicy"
                wsa:Action="http://.../ ws/PrizePaymentTeller/1.0/requests/PrizePaymentReportRequest "/>
                                                    …
  </wsdl:operation>
</wsdl:portType>
…
```

Figure 6. The protected element *wsdl:input* references the security policy where the security requirements it is related to are stated.

4.2 Security Policies Specification

One of the main principles of the PWSSec process is traceability. This traceability shows how security requirements are related to functional WS and to the security WS that address them. Security policies are defined with this objective in mind. In our security reference architecture the following policies are defined: *i*) Security Requirements policy stating what the security requirements are to be addressed in the information system. We limit our approach to those security requirements intended to secure the message's channel; *ii)* Security policy at Organization-level that all Security Zones should enforce. This type of security policies allows statements like "All WS provider agents that provide services to the Human Resources department will guarantee information's privacy according certain security policy", or, "All WS consumer agents should attach security mechanisms that guarantees the message's authentication"; *iii)* Security policy at Security Zone-level that should be applied by the 'Master' WSSecKern of the Security Zone (e.g.: all inbound messages should provide digital signatures so that message authentication and integrity can be guaranteed). This type of security policies allows statements like "All WS provider agents running in the CriticalSecurityZone shall verify that incoming messages attach security mechanisms that guarantees message reliability" or "All incoming messages targeted at WS providers running in the CriticalSecurityZone should provide mechanisms that guarantees non-repudiation"; *iv)* Security policy at Service-level that can be decompose into security policies defined by functional WS and security policies defined by security WS. The former will state what type of security requirements they are related to and how they should be addressed, and the latter will state what type of security requirements the security WS addresses (e.g.: authorization, message filtering or confidentiality)and what security mechanisms provides (e.g.: RBAC-based authorization service, addressing information-based filtering or symmetric ciphering).

In our case study we firstly defined a security policy that describes the security requirements of the software system at hand. This policy is directly derived from the security requirements elicited in the WSSecReq stage. In Figure 5, this security policy is depicted (for clarification's sake namespaces declarations have been omitted).

Security requirements have to be attached to the elements they protect. When elaborating and specifying security requirements for WS-based interactions we impose the following restrictions: *i)* every security requirement has to be related to one interaction between the WS consumer agent and the WS provider agent. This interaction will be assigned an action's name; *ii)* Every security requirement addresses just one type of security subfactor (i.e.: confidentiality, integrity, etc.); *iii)* Any WS-based security requirement intended to secure the message's channel will be related to one protected element. The possible set of elements that can be protected are specified in the WS-PolicyAttachment (version 1.0) specification of the WS-Policy framework [31] : wsdl:message, wsdl:message/wsdl:partwsdl:portType, wsdl:portType/wsdl:operation,

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy  xmlns:wsp="…" xmlns:wsse="..." xmlns:wsu="…">
        …
        <wsp:Policy wsu:Id=" PaymentPrizeInputMessagePolicy ">
        <SecurityRequirements product="P456">
                <SecurityRequirement>
                        <SecurityRequirementReference><Id>001013</Id></SecurityRequirementReference>
                </SecurityRequirement>
                <SecurityRequirement>
                        <SecurityRequirementReference><Id>001014</Id></SecurityRequirementReference>
                </SecurityRequirement>
                <SecurityRequirement>
                        <SecurityRequirementReference><Id>001018</Id></SecurityRequirementReference>
                </SecurityRequirement>
                <SecurityRequirement>
                        <SecurityRequirementReference><Id>001019</Id></SecurityRequirementReference>
                </SecurityRequirement>
        </SecurityRequirements>
</wsp:Policy>
```

Figure 7. This policy shows the security requirements related to the first *wsdl:input* protected element included in Figure 6.

wsdl:portType/wsdl:operation/wsdl:input, wsdl:portType/wsdl:operation/wsdl:output, wsdl:portType/wsdl:operation/wsdl:fault, wsdl:service and wsdl:service/port.

In our case study, the elements being protected correspond with the messages' input operations provided by the *WS-PrizePaymentTeller*. This association between the security requirements and the elements it protects is described in an independent policy. The element to be protected will reference the security policy where the security requirements are stated. In Figure 6, a fragment of this type of the security for the WS-PrizePaymentTeller is presented. It shows how the elements *wsdl:input* include the attribute *wsp:PolicyURIs* referencing the security policy that contains the security requirement they are related to (see Figure 6).

In our case study, we based the definition of the security policies on the 1.0 version (when this article was elaborated version 1.1 had just been delivered) of the WS-SecurityPolicy and WS-PolicyAttachment specifications. Every element in the WSDL description is associated, by means of a Policy URI reference, with its correspondent policy. Given the WSDL (for clarification's sake just portType section is shown) document describing the functional interface offered by the WS-PrizePaymentTeller depicted in Figure 7.

So far, we have defined what the security requirements are and what WSDL protected elements they are related to. Next, a security policy where every security requirement is associated with the security mechanism (s) that should address it is defined. This security policy will be defined from the point of view of the functional WS and will state what security mechanisms should be used for addressing a determined set of security requirements.

Before deploying the functional WS, the *CriticalSrvZone*'s Administrator registered in the WSSecKern (named *SportGambling-WSSecCriticalSrv*) the last three types of policies and its WSDL file.

In Figure 8, the *SecurityRequirements* elements specify the attribute *type*. The possible set of values of this attribute corresponds with the possible security subfactors or aspects to be taken into account in WS. Thus far we have defined the following potential values: *CommunicationsFiltering*, *CommunicationsIntegrity*, *CommunicationsConfidentiality*, *DataOriginAuthentication* and *ServiceAuthorization*. Every security WS (e.g.: Quality-of-Protection WS) covers just one of these types of security requirements.

4.3 Security Architecture Design

Once the set of security WS has been identified and the security policies have been defined, the next step is to design the security architecture. The elements defined in the security reference architecture should be instantiated and organized.The QoP WS shall be managed by a WSSecKern. This WSSecKern shall run within a Security Zone. So far, no WSSecKern has been defined so we define the Master WSSecKern of the Security Zone where the WS-PrizePaymentTeller will run. In our case study, the WS-based security reference architecture's identified elements are: Security Zone (Name: CriticalSrvZone, Criticality: *HIGH*, Administration: *Role CriticalWSZoneAdmin*); WS Security Kernel: *SportGambling-WSSecCriticalSrv*; WS-based Security Services: *Security XML-based firewall, Message Authentication, Message Integrity, Message Confidentiality and Authorization*; Functional WS: *WS-PrizePaymentTeller*; Security Policy at Organization-level: omitted; Security Policy at Security Zone-level: omitted; Security Policy at WSSecKern *SportGambling-WSSecCriticalSrv level*: omitted;Security Policy at business WS level: *WS-PrizePaymentTeller*; Security Policy at security WS level: *Quality-of-Protection WS*.

The WSSecKern will know what security WS there exist within its Security Zone and will associate one or more security WS's with one type of security requirement. In our case will associate the QoP WS with the following security requirement types: *CommunicationsIntegrity*, *CommunicationsConfidentiality*, *DataOriginAuthentication*.

```
<wsp:Policy  xmlns:wsp="…" wsu:Id="PaymentPrizeRequestSecRequirementSecMech">
        <wsp:SpecVersion wsp:Usage="wsp:Required"
                URI="http://schemas.xmlsoap.org/ws/2002/07/secext"/>
        <SecurityRequirements product="P456" type="CommunicationsConfidentiality">
                <SecurityRequirement>
                        <SecurityRequirementReference><Id>001018</Id></SecurityRequirementReference>
                <SecurityMechanism>
                <wsse:Confidentiality wsp:Usage="wsp:Required">
                                <wsse:Algorithm Type="wsse:AlgEncryption"
                                                URI="http://www.w3.org/2001/04/xmlenc#3des-cbc" />
                        <MessageParts>…</MessageParts>
                </wsse:Confidentiality>
                </SecurityMechanism>
                </SecurityRequirement>
        </SecurityRequirements>

        <SecurityRequirements product="P456" type="CommunicationsIntegrity">
                <SecurityRequirement>
                        <SecurityRequirementReference><Id>001013</Id></SecurityRequirementReference>
                </SecurityRequirement>
                <SecurityRequirement>
                        <SecurityRequirementReference><Id>001019</Id></SecurityRequirementReference>
                </SecurityRequirement>
                <wsse:Integrity wsp:Usage="wsp:Required">
                <wsse:Algorithm Type="wsse:AlgCanonicalization"  URI="http://www.w3.org/Signature/Drafts/xml-exc-c14n" />
                <wsse:Algorithm Type="wsse:AlgSignature"            URI=" http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
                <wsse:SecurityToken>
                                <wsse:TokenType>wsse:UsernameToken</wsse:TokenType>
                        <wsse:Claims>
                                        <wsse:SubjectName MatchType="wsse:Regexp">BankOrg\d{5}
                                        </wsse:SubjectName>
                                </wsse:Claims>
                </wsse:SecurityToken>
                        <MessageParts>…</MessageParts>
                </wsse:Integrity>
        </SecurityRequirements>
</wsp:Policy>
```

Figure 8. Security policy specifying the security mechanisms, and their parameters, to be used when addressing the security requirements.

Although we've omitted it, every WSSecKern in a Security Zone will register the meta-information associated to the security WS it manages. In addition, the Master WSSecKern will register the meta-information of all the WSSecKern, and of their correspondent security WS.

With all the previous information (functional WS's WSDL document and its security policies), the WSSecKern *SportGambling-WSSecCriticalSrv* will create a set of internal tables. When an incoming request targeted at the WS-PrizePaymentTeller is intercepted by the Master WSSecKern it will enforce the Organization-level, the SecurityZone-level and the WSSecKern-level policy in first place. Then, it will enforce the functional WS's security policy at the SOAP request-level. For simplification's sake, we will assume that only the last type of security policy has been defined.

Basically, when the WSSecKern intercepts an incoming request it will execute the following steps (notice that the same steps are applied for outgoing messages): i) The WSSecKern will obtain (see *a]* in Figure 9), the WS-Addressing Action included in the message's SOAP header (this a prerequisite for any incoming message to be allowed); ii) It will check that it supports that action, otherwise returns error; iii) Then, it will obtain the list of security requirements that applies to the request (see *b]* in Figure 9); iv) For every security requirement: a) Determines the security WS (and its responsible WSSecKern) that covers it within the Security Zone (see *d]* in Figure 9); b) Determines the type of WSDL element it protects (see *c]* in Figure 9); c) Fetches the security policy to be sent to the security WS that address that sort of security requirement (column *Parameters* in section *e]* of Figure 9). There could be more than one security policy each applying to different types of WSDL-protected element. The WSSecKern will have to fetch all of them and combine them in order to obtain the final policy [31]; d) Invokes the set of required security WS; e) Forward SOAP message to the ultimate recipient.

5. CONCLUSIONS AND FUTURE RESEARCH

In this paper we have presented the PWSSec process. PWSSec process allows developers to integrate security aspects when developing WS-based information systems from the very beginning of their development life-cycle. We have also introduced the stage WSSecArch of the PWSSec process intended to facilitate the allocation of WS-based security requirements into a WS-based security architecture. This stage defines a WS-based security reference architecture that specifies the set of security artifacts (e.g.: security zones, security WS, security policies, etc.)

a] Functional WS proxied by the WSSecKern.

WS-Addressing Action Request	*Web Service Recipient's Name*	*Destination Port*
http://…/PrizePaymentRequest	WS-PrizePaymentTeller	**<WSDL PORT FRAGMENT>**
http://…/PrizePaymentStateRequest	WS-PrizePaymentTeller	**<WSDL PORT FRAGMENT>**
http://…/PrizePaymentReportRequest	WS-PrizePaymentTeller	**<WSDL PORT FRAGMENT>**

b] Actions and security requirements associated.

WS-Addressing Action Request	*Product*	*Security Requirement*
http://…/PrizePaymentRequest	**456**	**001013**
http://…/PrizePaymentRequest	**456**	**001018**
http://…/PrizePaymentRequest	**456**	**001019**

c] Association between security requirements and its protected elements.

Product	*Security Requirement*	*Type*	*Protected Element*
456	**001013**	*DataOriginAuthentication*	**//wsdl:portType/wsdl:input**
456	**001018**	*CommunicationsConfidentiality*	**//wsdl:portType/wsdl:input**
456	**001019**	*CommunicationsIntegrity*	**//wsdl:portType/wsdl:input**

d] Association between security requirements types and security services that address them.

Security Requirement Type	*Security Service*	*Operation*
DataOriginAuthentication	**QoP**	**VerifyProtection**
CommunicationsConfidentiality	**QoP**	**VerifyProtection**
CommunicationsIntegrity	**QoP**	**VerifyProtection**

e] Security Requirement and Security Policies

Product	*Security Requirement*	*Element Protected*	*Parameters*
456	**001013**	**//wsdl:portType/wsdl:input**	**<security policy content>.**
456	**001018**	**//wsdl:portType/wsdl:input**	**<security policy content>.**
456	**001019**	**//wsdl:portType/wsdl:input**	**<security policy content>.**

Figure 9. Major internal tables generated by WSSecKern components.

that should be defined and how they interact. In addition, a case study where the PWSSec process and the WSSecArch are being applied has been presented.

Currently, several research lines are opened. We're refining the security reference architecture from the results obtained from the case study presented in this article. In addition, we're elaborating two more security WS: Security Token Service that abstracts the elements and mechanisms defined in the WS-Trust, XML Key Management System and Secure Assertion Mark-up Language standards; and a WS-based Authorization Service based on XACML standard and several research proposals [3, 5, 15, 16, 23, 32]. A tool that helps in the task of elaborating the security policies is also being studied [28].

6. ACKNOWLEDGMENTS

This research is part of the following projects: RETISTIC network (TIC2002-12487-E), of Dirección General de Investigación del Ministerio de Ciencia y Tecnología, and DIMENSIONS (PBC-05-012-1), financed by the FEDER and the "Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha.

7. REFERENCES

[1] C. J. Alberts, S. G. Behrens, R. D. Pethia, and W. R. Wilson, "OCTAVE Framework, Version 1.0", Carnegie Mellon. SEI. CMU/SEI-99-TR-017, September 1999.

[2] I. Alexander, "Misuse Cases: Use Cases with Hostile Intent", *IEEE Computer Software*, vol. 20, pp. 58-66, 2003.

[3] R. Bhatti, E. Bertino, A. Ghafoor, and J. B. D. Joshi, "XML-Based Specification for Web Services Document Security", *IEEE Computer*, vol. 37, pp. 41-49, 2004.

[4] R. Breu, K. Burger, M. Hafner, J. Jürjens, G. Popp, V. Lotz, and G. Wimmel, "Key Issues of a Formally Based Process Model for Security Engineering", Proc. 16th International Conference on Software and Systems Engineering and their Applications (ICSSEA'03), 2003.

[5] E. Damiani, S. D. C. d. Vimercati, S. Paraboschi, and P. Samarati., "A fine-grained access control system for XML documents", *ACM Transactions on Information and System Security (TISSEC)*, pp. 169-202, 2002.

[6] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo, and T. Newling, "Patterns: Service-Oriented Architecture and Web Services", *IBM Redbook*, 1st ed, 2004, pp. 345.

[7] E. B. Fernandez, "Two patterns for web services security", Proc. International Symposium on Web Services and Applications (ISWS'04), Las Vegas, NV, 2004.

[8] D. G. Firesmith, "Security Use Cases", *Journal of Object Technology*, vol. 2, pp. 53-64, 2003.

[9] D. G. Firesmith, "Engineering Security Requirements", *Journal of Object Technology*, vol. 2, pp. 53-68, 2003.

[10] D. G. Firesmith, "Common Concepts Underlying Safety, Security, and Survivability Engineering", SEI, Technical Note CMU/SEI-2003-TN-033, December 2003.

[11] C. Gutiérrez, E. Fernández-Medina, and M. Piattini, "Web Services Security: is the problem solved?" *Information Systems Security*, vol. 13, pp. 22-31, 2004.

[12] C. Gutiérrez, E. Fernández-Medina, and M. Piattini, "PWSSec: Process for Web Services Security", Proc. IEEE International Conference on Web Services 2005, Orlando, Florida, USA, 2005.

[13] C. Gutiérrez, E. Fernández-Medina, and M. Piattini, "Towards a Process for Web Services Security", Proc. WOSIS'05 en ICEIS'05, Miami, Florida, USA, 2005.

[14] IDC, 2005. See: http://www.idc.com/getdoc.jsp?containerId=prUS00190705].

[15] S. Indrakanti, V. Varadharajan, and M. Hitchens, "Authorization Service for Web Services and its Implementation", Proc. ICWS'04, San Diego, California, USA, 2004.

[16] H. Koshutanski and F. Massacci, "An Access Control Framework for Business Processes for Web Services", Proc. ACM Workshop on XML Security, 2003.

[17] P. Kruchten, *The Rational Unified Process: An Introduction*, 2nd. ed: Addison-Wesley Pub Co., 2000.

[18] A. v. Lamsweerde, "Elaborating Security Requirements by Construction of Intentional Anti-Models", Proc. 26th International Conference on Software Engineering, Edinburgh, 2004.

[19] J. D. Moffet and B. A. Nuseibeh, "A Framework for Security Requirements Engineering", Department of Computer Science, University of York, UK, Report YCS 368, August 2003.

[20] A. P. Moore, R. J. Ellison, and R. C. Linger, "Attack Modelling for Information Security and Survivability", Software Engineering Institute 2001.

[21] C. Nott, *Patterns: Using Business Service Choreography In Conjuction With An Enterprise Service Bus*, 2004.

[22] OASIS, "Web Services Security (WS-Security) - Specification 6 April 2004", 2004.

[23] OASIS, "eXtensible Access Control Markup Language (XACML) Version 2.0", 2005.

[24] OMG, "UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms", 2004.

[25] B. Schneier, "Attack Trees: Modeling Security Threats", *Dr. Dobb's Journal*, 1999.

[26] G. Sindre and A. L. Opdahl, "Eliciting Security Requirements with Misuse Cases", Proc. TOOLS-37'00, Sydney, Australia, 2000.

[27] IEEE Computer Society, "Software Engineering Body of Knowledge", 2004.

[28] M. Tatsubori, T. Imamura, and Y. Nakamura, "Best-Practice Patterns and Tool Support for Configuring Secure Web Services Messaging", Proc. 4th International Conference on Web Services (ICWS'04), San Diego, California, USA, 2004.

[29] A. Toval, J. Nicolás, B. Moros, and F. García, "Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach", *Requirements Engineering Journal*, vol. 6, pp. 205-219, 2001.

[30] D. Verdon and G. McGraw, "Risk Analysis in Software Design", in *IEEE Security & Privacy*, vol. 2, 2004, pp. 79-84.

[31] VeriSign, Microsoft, SonicSoftware, IBM, BEA, and SAP, "Web Services Policy Framework (WS-Policy)", 2004.

[32] R. Wonohoesodo and Z. Tari, "A Role based Access Control for Web Services", Proc. ICWS'04, San Diego, California, USA, 2004.

[33] WS-I, "Security Challenges, Threats and Countermeasures Versión 1.0", vol. 2005: WS-I, 2005.