Internet Research:

Electronic Networking Applications and Policy

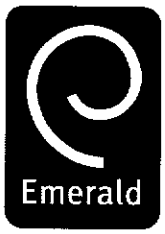
Security patterns and requirements for internet-based applications

David G. Rosado
ALARCOS Research Group, Information Systems and Technologies Department, UCLM-Soluziona Research and Development Institute, University of Castilla-La Mancha, Escuela Superior de Informática, Ciudad Real, Spain

Carlos Gutiérrez
ALARCOS Research Group, Information Systems and Technologies Department, UCLM-Soluziona Research and Development Institute, University of Castilla-La Mancha, Escuela Superior de Informática, Ciudad Real, Spain

Eduardo Fernández-Medina
ALARCOS Research Group, Information Systems and Technologies Department, UCLM-Soluziona Research and Development Institute, University of Castilla-La Mancha, Escuela Superior de Informática, Ciudad Real, Spain

Mario Piattini
ALARCOS Research Group, Information Systems and Technologies Department, UCLM-Soluziona Research and Development Institute, University of Castilla-La Mancha, Escuela Superior de Informática, Ciudad Real, Spain



Internet Research: Electronic Networking Applications and Policy, Vol. 16 No. 5, 2006,

Mission statement
Internet Research is a refereed journal that aims to describe, assess and foster understanding of the role of wide-area, multi-purpose computer networks, such as the Internet. The journal is international in scope and both inter-disciplinary and cross-disciplinary.

ISBN-13 978-1-84663-232-7
ISBN-10 1-84663-232-3
ISSN 1066-2243

Awarded in recognition of Emerald's production department's adherence to quality systems and processes when preparing scholarly journals for print

Internet Research
is indexed and abstracted in:

BUBL
Cabell's Directory of Publishing Opportunities in Management & Marketing
CompuMath Citation Index
Computer Science Index
Current Contents/Engineering Computing & Technology
Current Awareness Abstracts
Emerald Review (formerly Anbar)
ERIC Clearinghouse on Information Technology
Fulltext Sources Online
Information Science Abstracts and Fulltext Sources Online
INSPEC Database
Library and Information Science Abstracts (LISA)
Library Literature
PAIS Indexes
Research Alert
SciSearch®
Science Citation Index Expanded
Scopus

This journal is also available online at:
Journal information
www.emeraldinsight.com/intr.htm
Table of contents
www.emeraldinsight.com/1066-2243.htm

Internet services available worldwide at www.emeraldinsight.com

Emerald Group Publishing Limited
60/62 Toller Lane, Bradford
BD8 9BY, United Kingdom
Tel +44 (0) 1274 777700
Fax +44 (0) 1274 785200
E-mail **information@emeraldinsight.com**

INVESTOR IN PEOPLE

Regional offices:

For North America
Emerald, 875 Massachusetts Avenue, 7th Floor, Cambridge, MA 02139, USA
Tel Toll free +1 888 622 0075; Fax +1 617 354 6875
E-mail **america@emeraldinsight.com**

For Japan
Emerald, 3-22-7 Oowada, Ichikawa-shi, Chiba, 272-0025, Japan
Tel +81 47 393 7322; Fax +81 47 393 7323
E-mail **japan@emeraldinsight.com**

For Asia Pacific
Emerald, 7-2, 7th Floor, Menara KLH, Bandar Puchong Jaya, 47100 Puchong, Selangor, Malaysia
Tel +60 3 8076 6009; Fax +60 3 8076 6007
E-mail **asiapacific@emeraldinsight.com**

For China
Emerald, 12th Floor, Beijing Modern Palace Building
No. 20, Dongsanhuan Nanlu, Chaoyang District, Beijing 100022, China
Tel +86 (0) 10 6776 2231; Fax +86 (0) 10 6779 9806
E-mail **china@emeraldinsight.com**

Customer helpdesk:
Tel +44 (0) 1274 785278; Fax +44 (0) 1274 785204;
E-mail **support@emeraldinsight.com**
Web **www.emeraldinsight.com/customercharter**

Orders, subscription and missing claims enquiries:
E-mail **subscriptions@emeraldinsight.com**
Tel +44 (0) 1274 777700; Fax +44 (0) 1274 785200

Missing issue claims will be fulfilled if claimed within four months of date of despatch. Maximum of one claim per issue.

Reprints service:
Tel +44 (0) 1274 785135
E-mail **reprints@emeraldinsight.com**
Web **www.emeraldinsight.com/reprints**

Permissions service:
Tel +44 (0) 1274 785139
E-mail **permissions@emeraldinsight.com**
Web **www.emeraldinsight.com/permissions**

Emerald is a trading name of Emerald Group Publishing Limited

Printed by Printhaus Group Ltd, Scirocco Close, Moulton Park, Northampton NN3 6HE

The current issue and full text archive of this journal is available at
www.emeraldinsight.com/1066-2243.htm

Security patterns and requirements for internet-based applications

David G. Rosado, Carlos Gutiérrez,
Eduardo Fernández-Medina and Mario Piattini
ALARCOS Research Group, Information Systems and Technologies Department, UCLM-Soluziona Research and Development Institute, University of Castilla-La Mancha, Escuela Superior de Informática, Ciudad Real, Spain

Abstract

Purpose – The purpose of this paper is that of linking security requirements for web services with security patterns, both at the architectural and the design level, obtaining in a systematic way a web services security software architecture that contains a set of security patterns, thus ensuring that the security requirements of the internet-based application that have been elicited are fulfilled. Additionally, the security patterns are linked with the most appropriate standards for their implementation.

Design/methodology/approach – To develop secure WS-based applications, one must know the main security requirements specified that applications have to fulfil and find appropriate security patterns that assure, through combination or relationships between them, the fulfilment of the implicated security requirements. That is why a possible link or connection between requirements and patterns will have to be found, attempting to select for a determined security requirement the best security patterns that solve this requirement, thus guaranteeing the security properties for internet-based applications.

Findings – Using security patterns, that drive and guide one towards a secure development as well as towards security software architecture, one can be sure that this design based on these patterns fulfils and guarantees the most important security requirements of the internet-based applications through the design and implementation of security solutions that provide reliable security services.

Practical implications – Security architecture for internet-based applications and web services can be designed considering the security requirement types that it must fulfil and using the most appropriate security patterns.

Originality/value – This paper proposes a relationship between security requirements that can be specified for internet-based applications and the possible security patterns that can be used in the design and implementation of the secure system based on the internet, guaranteeing that these security requirements are fulfilled.

Keywords Worldwide web, Security products, Computer applications

Paper type Research paper

Introduction

Web services are a natural consequence of the evolution of the internet. Since its beginnings as a way to share and distribute information on a global scale, effectively

This research is part of the following projects: DIMENSIONS (PBC-05-012-2) and MISTICO (PBC06-0082) financed by FEDER and by the "Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha" (Spain), y CALIPO (TIC2003-07804-C05-03) granted by the "Dirección General de Investigación del Ministerio de Ciencia y Tecnología" (Spain).

Internet Research
Vol. 16 No. 5, 2006
pp. 519-536

1066-2243
DOI 10.1108/10662240610710996

becoming a giant distributed content library, the internet has been progressively widening its reach to enable more sophisticated forms of interaction between browser clients and servers: single form-based interactions, retail ecommerce applications, and more complex business-to-business interactions (Curbera *et al.*, 2001). WS-based software has evolved from a state of creation to more rampant consumption, according to an IDC report, which expects the market to reach around $15 billion by 2009 (IDC, 2005).

The benefits of having a loosely-coupled, language-neutral, platform-independent way of linking applications within organizations, across enterprises, and across the internet are becoming more evident as web services are used in pilot programs and in large-scale production. Moving forward, our customers, industry analysts, and the press identify a key area that needs to be addressed as web services become more mainstream: security. A key benefit of the emerging web services architectures is the ability to deliver integrated, interoperable solutions. Therefore, ensuring the integrity, confidentiality and availability of web services through the application of a comprehensive security model is critical, both for organizations and their customers (IBM, 2002).

Ongoing advances in technology and the growth of the internet are introducing not only an increase in the number of vulnerabilities being found, but also an increase in the complexity of system administration, incident handling and forensic analysis work. There have been progressive changes in intruder techniques, increased difficulty of detecting an attack, increased amounts of damage, and an increased difficulty in catching the attackers (Fox, 2001). Security is a key factor for companies when adopting web services in their mission critical business transactions. Without taking care of security, companies could not use web services in the insecure internet environment. As the statistics presented by the CERT show, the number of incidents related to security has exponentially grown during the last years (they have passed from 3.734 incidents reported in 1998 to 137.529 in 2003; Total incidents reported (1988-2003): 319.992) (CERT, 2006).

To secure web services, standard bodies such as the World Wide web Consortium (W3C), the Organization for the Advancement of Structured Information Standards (OASIS), the Liberty Alliance, and others are developing XML-based security standards to solve security problems (Imamura and Tatsubori, 2003). This diversity, also found in the context of WS security (Gutiérrez *et al.*, 2004) has made us consider their application, from a global perspective, as a very complex and hard process to understand with a very difficult learning curve. At present, there is an initiative that offers a methodological approach to elicit security requirements as well as to construct security architectures for WS-based systems, the PWSSec (Process for web Services Security) process (Gutiérrez *et al.*, 2005a). In this paper we complement that approach with a mechanisms to define traceability both in direct and reverse engineering from (or to) the security requirements to (of from) the security standards that implement these requirements, building in the middle a security architecture, which offer a high level design solution to the specified requirements.

One of the problems we face in everyday practice is that we want to secure an application without spending excessive time and effort; for this reason, we are tempted to use some known solutions like putting up a firewall or using simple password authentication. Applying a pattern, a solution that has been already extensively used in practice, might seem to be a reasonable idea. In many cases, however, a solution

applied without a thorough understanding of security requirements does not provide adequate protection within the specific context (Kis, 2002).

In this paper, we will study the most important security requirement types that we can find in the literature but we have adapted them to WS. Also, using existing security patterns, we will select a set of them that we consider more interesting for WS and we will attempt to relate them to some security requirements specified covered by these security patterns. A typical collection of security patterns covers a wide range of topics and there is often no obvious connection between them. There is no process for using the patterns that will connect them together. We are aimed at establishing a link between security requirements and patterns, indicating a possible technology with security standards for WS.

We can find in the literature much research in the area of web services security, both regarding security requirements analysis (Firesmith, 2003; Firesmith, 2004; Gutiérrez *et al.*, 2005b; OASIS, 2005; W3C, 2004; WS-I, 2005; Yagüe *et al.*, 2005; Yagüe and Troya, 2002) and concerning the definition of security patterns (Alexander *et al.*, 1977; Buschmann, 1999; Buschmann *et al.*, 1996; Fernandez, 2002; Fernandez and Pan, 2001; Fernandez *et al.*, 2003; Lehtoren and Pärssinen, 2002; Ramachandran, 2002; Rosado *et al.*, 2006; Steel *et al.*, 2005; Yoder and Barcalow, 1997). However, there are no approaches connecting security requirements with security patterns, so there are no traceability mechanisms allowing choosing the most appropriate security pattern depending on a specific security requirement. Moreover, those approaches do not offer a way to select the most appropriate technology that better implements these patterns.

The rest of the paper is organized as follows: in next section, we will show the most important security requirement types for web services that we have taken into consideration. Later, we will define security patterns (design and architecture), and we will analyze some of the most important patterns, regarding with web services security. In the next section we will first study the connection between security requirement types, patterns that fulfil these requirements and possible technologies relevant for these patterns, and we will finish stating a pattern example. Finally, we will put forward our conclusions and the future work.

WS security requirements

We can find in the literature the definition of several security requirement types but in this work we will focus on those that are considered to be more important when we work with internet-based applications, specifically with web services.

These generic security requirements (Firesmith, 2004; OASIS, 2005; W3C, 2004; WS-I, 2005) can be "redefined" according to the special characteristics of web services-based applications as follows:

- *Authentication:* Authentication in distributed systems can be divided into two processes: Entity authentication: consisting of verifying either the identity claimed by a WS consumer agent or that of the principal using the WS consumer agent (or both); or message authentication: consisting of surely knowing that a certain message is genuine, in other words, that it comes from the source that produced it and without any modification.
- *Authorization:* In the context of WS, authorization basically consists of guaranteeing that only the proper WS agents consumers or principals, and under suitable circumstances, can access to the services offered by a certain WS

provider agent. The reference standard in this sense is XACML (eXtensible Access Control Mark-up Language).

- *Confidentiality:* Confidentiality in WS is mainly considered in relation to the content of the exchanged messages. The most common way used to guarantee message confidentiality is by using cryptography. Keeping the information exchanged among web services nodes secret is another of the main properties that should be guaranteed in order to consider the channel secure.
- *Integrity:* Integrity in the context of WS is mainly related to the integrity of the messages that are exchanged among WS. This property guarantees that the information received by a web service remains the same as the information that was sent from the client. A simple checksum might offer integrity when accidental changes are made to data.
- *Privacy:* The main purpose of privacy in WS consists of guaranteeing the right custody (that is the right divulgation) of the sensitive information stored by a certain organization about its customers. The complete infrastructure about privacy has been already established by the W3C consortium publishing the P3P (Policy for Privacy Preferences) recommendations APPEL (P3P Preference Exchange Language) (Cranor *et al.*, 2002) and EP3P o EPAL (Ashley *et al.*, 2002).
- *Non-repudiation:* In the web services world, it is necessary to be able to prove that a client utilized a service (requester non-repudiation) and that the service processed the client request (provider non-repudiation). This security issue is covered by means of digital signatures.
- *Availability:* The need to take care of the availability aspects for preventing denial-of-service attacks or to arrange redundancy systems is a crucial point in web services technology, especially, in those scenarios in which the services provide critical services: real-time services, certification revocation lists services, etc.
- *Security auditing:* A service that reliably and securely records security-related events producing an audit trail enabling the reconstruction and examination of a sequence of events. Security events could include authentication events, policy enforcement decisions, and others. The resulting audit trail may be used to detect attacks, confirm compliance with policy, detect abuse, or for other purposes.
- *Perimeter security:* You can secure your network perimeter and information resources with our market-leading perimeter security solutions. They ensure that the right people can access your network resources as well as they detect and thwart attacks to make sure the wrong people cannot access.
- *Trust management:* Trust management is an approach to access control whereby access is granted based on trust established in a negotiation between the service requester and the service provider. In this negotiation, credentials – signed assertions that describe the owner's attributes – are iteratively exchanged to build trust between the negotiation participants. Credentials are typically based on standards such as X.509v3 (Housley *et al.*, 1999), simple public key infrastructure (SPKI) (Ellison *et al.*, 1999), OASIS Security Assertion Markup Language (SAML) (OASIS, 2003), W3C XML Key Management System (Ford *et al.*, 2001), or IBM & Microsoft WS-Trust (Anderson *et al.*, 2005).

- *Delegation:* Delegation is an approach where an entity provides other entities with all or some of its privileges or rights. This is considered a useful and effective method to enhance the scalability of a distributed system and decentralize access control tasks. The reference standard WS-Trust (Anderson *et al.*, 2005), and SAML v2.0 provide a mechanism for delegation.
- *Federation:* It is virtually impossible to rely on a universal point of control for identity information. In some cases, companies have multiple identity repositories for their applications, thus creating a corporate infrastructure fragmented into silos of activities. In addition, when companies do business with each other, they need to exchange information about their respective users in a trusted way. Companies involved in identity federation establish trusted relationships, allowing their respective users to access resources hosted by a business partner. In this case, companies issue security tickets to their users that can be processed by relying parties.
- *Messaging reliability:* In the web services world, the sender's messaging component can tolerate network failures by repeatedly sending a message until it is acknowledged by the receiver's messaging component; this interaction can occur even after the sending process has terminated (or is otherwise unavailable). The receiver's messaging component can tolerate the unavailability of the receiving process by maintaining messages until the receiving process is ready. The reference standard of OASIS WS-ReliableMessaging (Bilorusets *et al.*, 2004), and the specifications WS-Reliability (Evans *et al.*, 2003) and HTTPR (Todd *et al.*, 2001) provide us with a reliable messaging framework about protocols and open networks that can be used.

WS security patterns

In this section, we will define what security patterns are (architectural patterns and design patterns) and we will study some of those that are more relevant in the area of web services security.

Security pattern definition

A security pattern describes a particular recurring security problem that arises in specific contexts and presents a well-proven generic scheme for its solution. According to their level of abstraction, patterns can be divided into architectural patterns (high-level) and design patterns (mid-level) (Buschmann *et al.*, 1996).

The benefits of using patterns are: they can be revisited and implemented at anytime to improve the design of a system; less experienced practitioners can benefit from the experience of those more fluent in security patterns; they provide a common language for discussion, testing and development; they can be easily searched and categorized; they provide reusable, repeatable and documented security practices; they do not define coding styles, programming languages or vendors (Berry *et al.*, 2002).

In the context of WS we envisage that a security pattern-based approach should be followed, at both the architectural and the design level, in order to facilitate the construction of such systems as well as guarantee its reutilization across different projects. Today, software architects can apply many of the existing design patterns to web services. The use of these patterns can greatly help architects build scalable, reliable, and robust web services architectures.

Architectural patterns. An architectural pattern is a description of the elements and the relationship types together with a set of constraints on how they may be used. A pattern can be thought of as a set of constraints on an architecture – on the element types and their patterns of interaction – and these constraints define a set or family of architectures that satisfy them. For example, client-server is a common architectural pattern. An architectural pattern is not an architecture but it still conveys a useful image of the system – it imposes useful constraints on the architecture and, in turn, on the system (Bass *et al.*, 2003).

An architectural pattern expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them (Buschmann *et al.*, 1996). An architectural pattern is a high-level abstraction. The choice of the architectural pattern to be used is a fundamental design decision in the development of a software system. It determines the system-wide structure and constrains the design choices available for the various subsystems. It is, in general, independent of the implementation language to be used.

One of the most useful aspects of patterns is that they exhibit known quality attributes. This is why the architect chooses a particular pattern and not one at random. Some patterns represent known solutions to performance problems, others lend themselves well to high-security systems, still others have been successfully used in high-availability systems. Choosing an architectural pattern is often the architect's first major design choice (Bass *et al.*, 2003).

Design patterns. Mature software design patterns, like patterns in any other discipline, capture solutions that have been developed and evolved over time. Hence, they are not the designs that people tend to generate initially. Mature patterns reflect much iteration of untold redesign and recoding, as developers have struggled for greater reuse and flexibility in their software. Design patterns capture refined solutions in a succinct and easily applied form.

A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context (Buschmann *et al.*, 1996). A design pattern is a mid-level abstraction. The choice of a design pattern does not affect the fundamental structure of the software system, but it does affect the structure of a subsystem. Like the architectural pattern, the design pattern tends to be independent of the implementation language to be used.

Collection of security patterns

Once we have selected a set of security requirements of all aforementioned that we have considered them interesting for our study, our aim is to identify, for these set of requirements, the security patterns that guarantee, in some way, one or several security requirements types, that is, for each security requirement type our objective is to identify one or several patterns that ensure the mentioned requirement.

Using security patterns, which drive and guide us towards a secure development as well as towards security software architecture, we can be sure that our design based on these patterns fulfils and guarantees these security requirements through the design and implementation of security solutions that provide reliable security services. In the next paragraphs, we will briefly describe some of the most important patterns found in

the literature related to web services. For further information about these patterns, please see the shown references:

- *Audit interceptor:* This pattern (Steel *et al.*, 2005) works in conjunction with the Secure Logger pattern and provides instrumentation of the logging aspects in the front. Besides, this pattern enables the administration and manages the logging and audit in the back-end. Figure 1 depicts the class diagram for the audit interceptor pattern. The *Client* attempts to access the *Target*. The *AuditInterceptor* class intercepts the request and uses the *AuditEventCatalog* to determine if an audit event should be written in the *AuditLog*.
- *Secure logger:* This pattern (Steel *et al.*, 2005) defines how to capture the application-specific events and exceptions in a secure and reliable manner to support security auditing.
- *Message interceptor* (Steel *et al.*, 2005) checks for and verifies the quality of XML message-level security mechanisms, such as XML Signature and XML Encryption in conjunction with a security token. In addition, this pattern helps us verify and validate applied security mechanisms in a SOAP message when processed by multiple intermediaries (actors). It supports a variety of signature formats and encryption technologies used by these intermediaries.
- *Assertion builder:* This pattern (Steel *et al.*, 2005) defines how an entity assertion (for example, authentication assertion or authorization assertion) can be built.
- *Secure pipe:* This pattern (Steel *et al.*, 2005) shows how to secure the connection between client and server, or between servers when connecting between trading partners. In a complex distributed application environment, there will be a mixture of security requirements and constraints between clients, servers, and any intermediaries. It adds value by requiring mutual authentication and establishing confidentiality or non-repudiation between trading partners. This is particularly critical for B2B integration using web services.
- *Secure message router* (Steel *et al.*, 2005) facilitates secure XML communication with multiple partner endpoints that adopt message-level security and identity-federation mechanisms (Figure 2). It acts as a security intermediary component that applies message-level security mechanisms to deliver messages

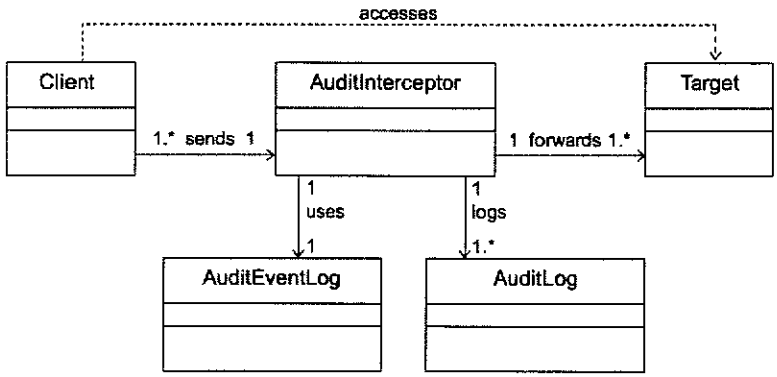


Figure 1.
Audit interceptor class diagram

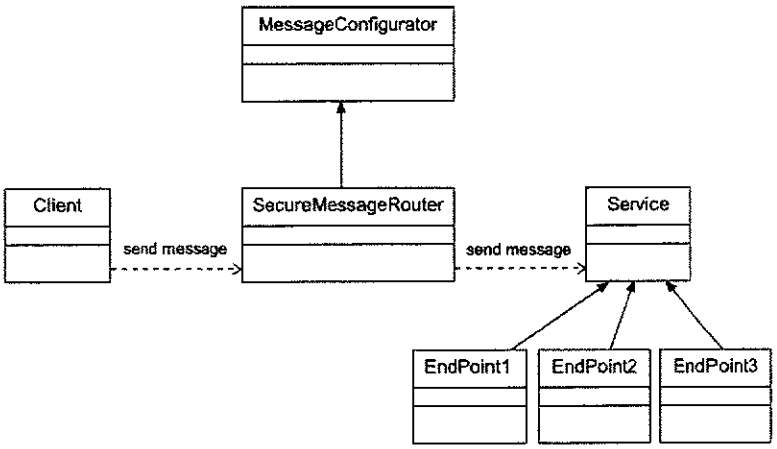


Figure 2.
Secure message router class diagram

to multiple recipients where the intended recipient would be able to access only the required portion of the message and remaining message fragments would be made confidential.

- *Authoritative source of data*: This pattern (Romanosky, 2002) is used to verify the validity of data and their origin. It prevents the system from using outdated and incorrect information and reduces the potential risk of processing and propagating fraudulent data.
- *Credential tokenizer* (Steel *et al.*, 2005) describes how a principal's security token can be encapsulated, embedded in a SOAP message, routed, and processed. Figure 3 depicts a class diagram of the credential tokenizer. The credential tokenizer can be used to create different security tokens (*SecurityToken*),

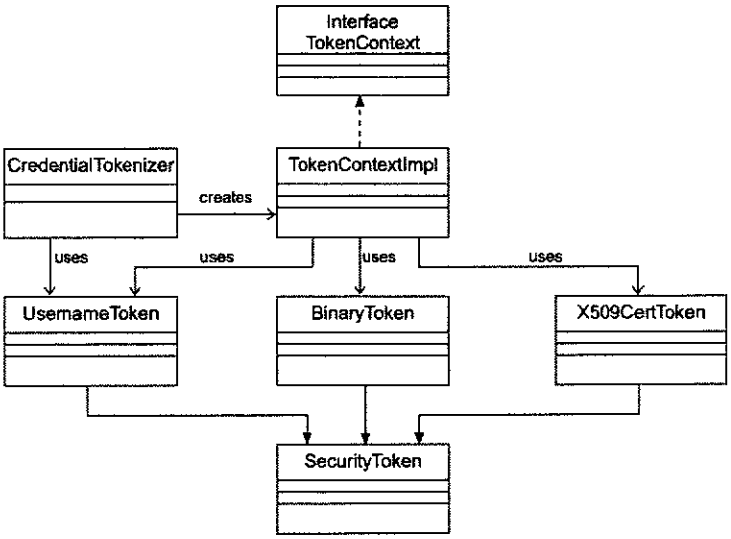


Figure 3.
Credential tokenizer class diagram

including username token, binary tokens and X.509v3 certificate. When creating a security token, the credential tokenizer creates a system context (*TokenContext*) that encapsulates the token type, the name of the principal, the service configuration, and the protocol binding that the security token supports.

- *Single sign-on (SSO) Delegator* (Steel *et al.*, 2005) describes how to construct a delegation agent for handling a legacy system for single sign-on (SSO).
- *Layered security*: This pattern (Romanosky, 2001) is aimed at dividing a system's structure into several layers to improve the security of the system by securing all of the layers. One major drawback of using this pattern is that it increases complexity at the architecture level.
- *Check point*: This pattern (Cheng *et al.*, 2003; Yoder and Barcalow, 1997) centralizes and enforces the security policy and encapsulates the algorithm to put the security policy into operation. The algorithm can contain any number of security checks. This pattern can be also used to keep track of the failed attempts of security breaches, which helps us take appropriate action if the failures are malicious activities.
- *Data filter*: This pattern (Flanders and Fernandez, 1999) filters the contents of client requests in a distributed system, according to predefined policies. Filtering can occur locally or remotely. In many distributed systems, e.g. the internet, requests for services or data need to be filtered according to institution policies, legislative restrictions, privacy needs, etc.
- *Authenticator*: The Authenticator pattern (Lee Brown *et al.*, 1999) describes a general mechanism for providing identification and authentication to a server from a client. It has the added feature of allowing protocol negotiation to take place using the same procedures. The pattern operates by offering an authentication negotiation object which then provides the protected object only after authentication is successful.
- *BodyGuard*: This pattern (Das Neves and Garrido, 1998) allows us to share objects and control their access in a distributed environment without system level support for distributed objects. The Bodyguard is a pattern that simplifies the management of object sharing over a network. It provides message dispatching validation and assignment of access rights to objects in non-local environments, to prevent the incorrect access to an object in collaborative applications.

Relationship between security requirements, patterns and technologies

For establishing the relationship or traceability from a security requirement type to a security pattern or patterns, first of all, we have selected a set of security requirements of those aforementioned, a set of security patterns that we have considered the most important and interesting for internet-based applications from those found in the literature and that fulfil and guarantee the security requirements specified, and also we have studied some security standards and specifications for WS related to these patterns, that we can see in Table I.

As we have commented before, we can find in the literature many security requirements and much more security patterns, so we show in Table I just a summary of the analysis we have performed, considering only some of the most relevant requirements and patterns.

Security requirements	Architectural patterns	Design patterns	Security standards
Authentication	QoP Role-based security Assertion coordinator Data filter Check point SSO Cryptographic Direct authentication	Assertion builder SSO Delegation Sender authentication Authenticator Credential tokenizer	LibAlliance SASL based authentication service; SAML 2.0WS-Security +SAML 2.0 + Kerberos Token ProfileXML Key Management SystemWS-Security + XML Digital Signature SAML 2.0 + WS-Security + SAML Token Profile + XML Digital Signature SAML 2.0, Liberty Alliance Project ID-FF 1.1, WS-Federation
	Brokered authentication	Security token service, X.509 PKI, Kerberos	WS-Security +SAML 2.0 + Kerberos Token Profile
Authorization	PEP + PDP + PRP + PIP + PAP Data Filter Bodyguard Check point, Firewall	XML firewall filter Assertion builder Authorization RBAC Session	WS-Policy + WS-SecurityPolicy; XACML Profile; XrML ODRLWS-Authorization
Confidentiality	QoP, Encryption, Cryptographic, Layered security	Message inspector, Information secrecy, Secure pipe, Session	WS-Security + XML Encryption
Integrity	QoP Firewall Data filter Layered security	Message inspector, Secure pipe, Message integrity, Secure message router, Authoritative source of data multilevel security	WS-Security + XML Digital Signature
Audit	Check point Single Access point	Audit interceptor Secure logger	

Table I.
Relationship between requirements, patterns and standards

If we select a security requirement type from the table (i.e. "Authentication"), we can see clearly in Figure 4 the relationship existing between this security requirement type, the patterns and standards related. We know some patterns for the requirement "Authentication" as the architectural pattern *Brokered Authentication* where the web service validates the credentials presented by the client, without the need of a direct relationship between the two parties, or the architectural pattern *Direct Authentication* where the web service acts as an authentication service to validate credentials from the client (see Figure 4). The *Brokered Authentication* pattern can be decomposed into three specialized design patterns where each type uses different mechanisms to authentication broker between a client and a service; the security token service pattern when the mechanism used is a web service

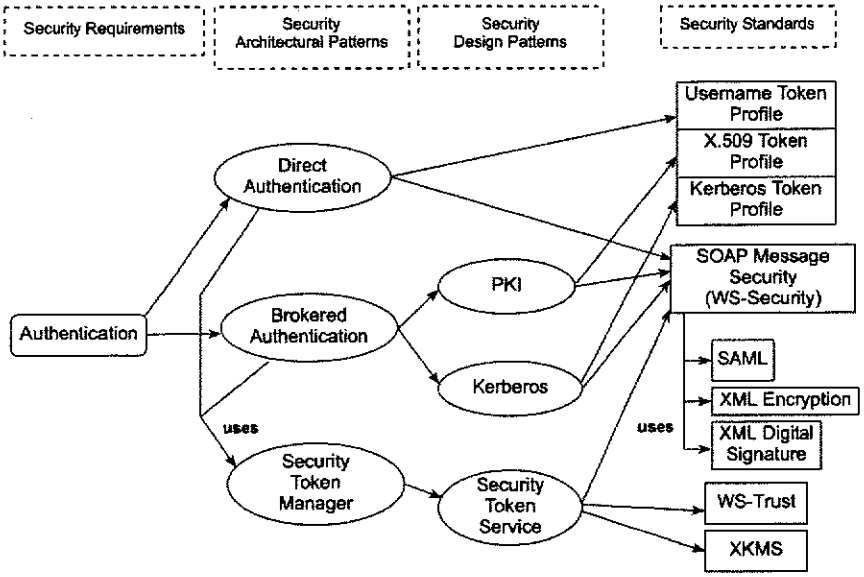


Figure 4.
Relation between requirements, patterns and standards for authentication requirement

security token service (STS), the PKI pattern that uses the X.509 token profile standard and the Kerberos pattern that uses the Kerberos protocol to verify the credentials presented by the client application. Therefore, if we use these security patterns, we will be able to develop and design the architecture and the mechanisms that achieve and fulfil the desired requirement type taking into account the infrastructure to use.

For the rest of security requirements we can make the same study looking for the security patterns that better cover the requirements and relating them to the standards and specifications. However, we have only shown an example with the security requirement "Authentication" to demonstrate the relationship between security requirements, patterns and technologies that we can obtain. Thus, message integrity is provided by XML Signature in conjunction with security tokens to ensure that modifications to messages are detected. The integrity mechanisms are designed to support multiple signatures, potentially by multiple SOAP roles, and to be extensible to support additional signature formats. Message confidentiality leverages XML Encryption in conjunction with security tokens to keep portions of a SOAP message confidential, and so on.

As for the standards and specifications, regarding this example, the most important thing that we must know is that an X.509 certificate specifies a binding between a public key and a set of attributes that includes (at least) a subject name, issuer name, serial number and validity interval. An X.509 certificate may be used to validate a public key that may be used to authenticate a SOAP message or to identify the public key with a SOAP message that has been encrypted (OASIS, 2006a). Username Token Profile specification describes how a web service consumer can supply a *UsernameToken* as a means of identifying the requestor by "username", and optionally using a password (or shared secret, or password equivalent) to authenticate that identity to the web service producer (OASIS, 2006b). Kerberos Token Profile

specification defines how to encode Kerberos tickets and attach them to SOAP messages. In addition, it specifies how to add signatures and encryption to the SOAP message, in accordance with WSS: SOAP Message Security, which uses and refers to the Kerberos tokens (OASIS, 2006c).

Example: quality of protection "QoP" pattern

We have defined the "QoP" Security Pattern, abstracted from the mechanisms defined in the WS-Security specification (OASIS, 2006d), which addresses "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. The WS-Security standard is the "de facto" WS security standard that implements this security service. In our example, we have applied the WS "QoP" architectural pattern in order to address these security requirements: data origin authentication, integrity and confidentiality.

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 express the type of security requirement that its instance covers (message authentication & integrity, confidentiality), the specific security mechanisms to be used and the WS specification that it deploys (Gutiérrez *et al.*, 2005c).

Following with our proposal to relate security requirements to security patterns, we can see in Figure 5 this relationship for the "QoP" pattern. This pattern covers confidentiality, data origin authentication and integrity at the message level, and it is composed by two security design patterns: "*Data Confidentiality*" that is destined to protect sensitive data contained in a message using encryption (XML Encryption) and "Data Origin Authentication & Integrity" that verifies that messages have not been tampered with in transit (data integrity) and that they originate from the expected sender (authenticity), using certificates, tokens and signatures (SAML, XML Digital Signature, WS-Trust, etc.) making use of the "Security Token Manager" pattern that

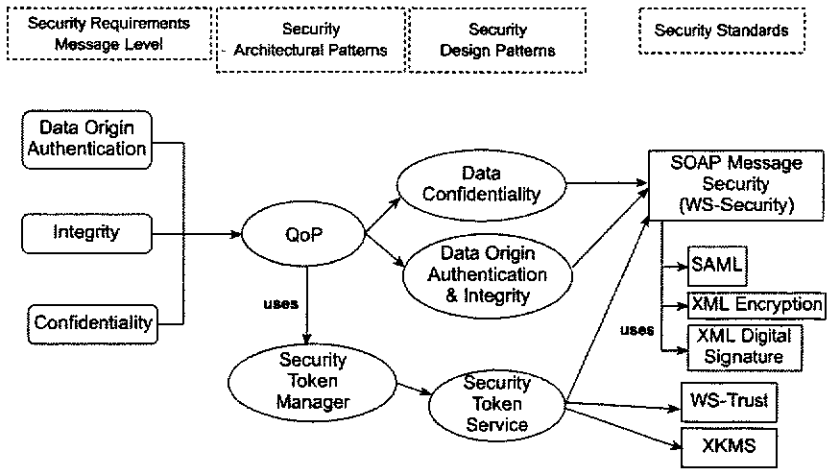


Figure 5.
Relation between requirements, patterns and standards for "QoP" pattern

manages more appropriate security tokens depending of the infrastructure used (PKI, Kerberos, etc.).

We can see in Figure 6 the UML diagram of the subsystem and interfaces for this pattern, indicating the main relationships to others subsystems and interfaces that this pattern can use to carry out its function of quality of protection.

All specifications and standards appearing in Figure 6 are variability points depending on the number and type of keys used, on the type of encryption used and on the key management infrastructure in use. For example, the type of encryption can be symmetric, both the sender and recipient share a key that is used to perform both encryption and decryption, and the most common algorithms include Rijndael (AES) and Triple DES (3DES), or it can be asymmetric, the sender encrypts data with one key,

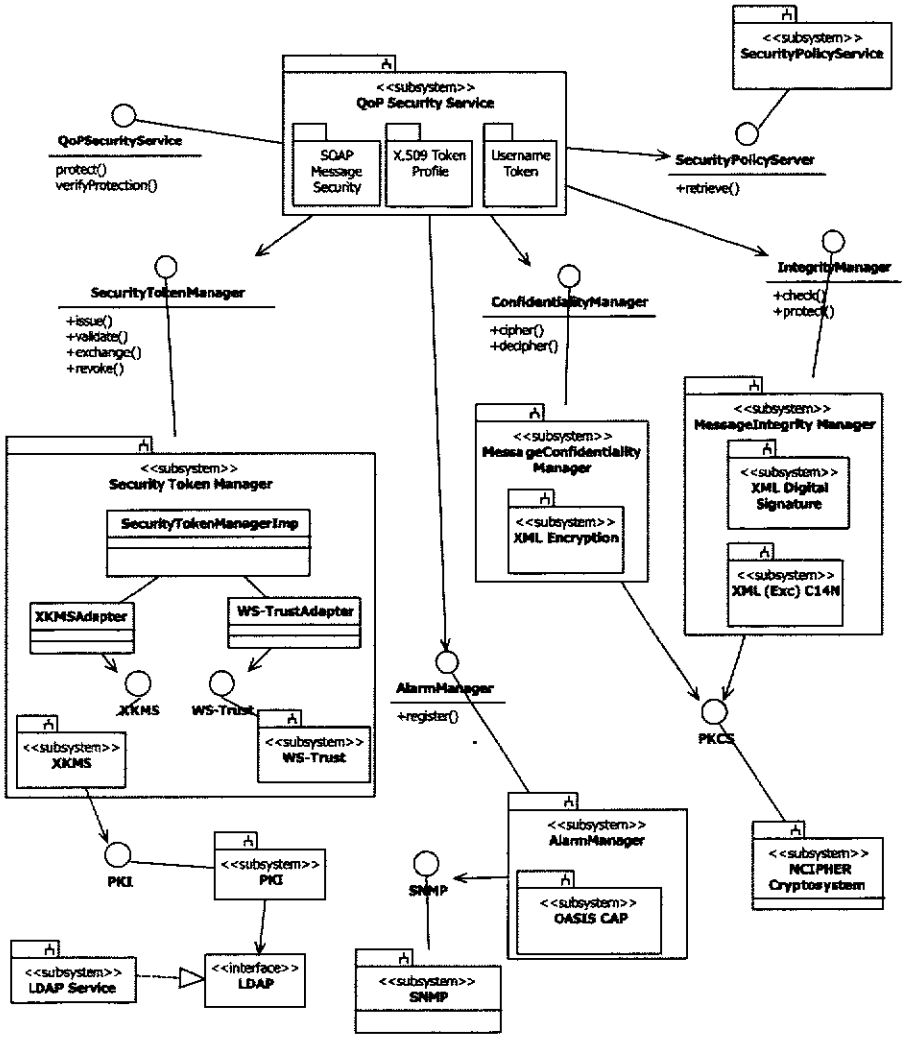


Figure 6.
Relations between subsystems and interfaces of the "QoP" pattern

and the recipient uses a different key to decrypt cipher text, and the most commonly used asymmetric algorithm is the RSA algorithm. The type of authentication that we can use is the Username Token, Kerberos Token and X.509 PKI.

First of all, we have the subsystem "QoP" Security Service that offers the service of Quality of Protection to the system. Each subsystem must implement an interface that is used by the rest of elements that need the service that the owner subsystem of the interface offers. This subsystem implements the interface "*QoPSecurityService*" with the methods *protect* and *verifyProtection*, as we can see in Figure 6. These two methods are used by the elements for applying the service to outbound/inbound messages according to the policy established for this service. This subsystem has relationships with others subsystems or interfaces, such as the relationship to the subsystem *Security Token Manager*, through its interface "*SecurityTokenManager*" that manages security tokens and can be implemented using WS-Trust, XKMS with PKI infrastructure, etc., established in its security policy; it is related to the subsystem *Message Confidentiality Manager*, through its interface "*ConfidentialityManager*" that ciphers or deciphers the message offering the message confidentiality service using XML Encryption; and it is related to the subsystem *Message Integrity Manager*, through its interface "*IntegrityManager*" that checks and protects the message offering the message integrity service using XML Digital Signature. Also, it can be related to the subsystem Security Policy Service, this is optional, where it would manage all policies associated with the services offered by the system. We have stated that this is optional because policies can be managed and implemented into the own subsystem (i.e. "QoP" subsystem) without having to be related to this subsystem.

Other possible relationship, non compulsory, is the relationship between "QoP" Security Service subsystem and an Alarm subsystem, using a common protocol of alarms (CAP, Common Alerting Protocol) establishing an alarm system in the application, communicating elements with others, indicating an event or an alarm generated and sent to others subsystems, for example, when an user wants to login and uses resources and services but he/she is not authorized to do it. The system must create a register of the vain attempt, unauthorized access, and so on.

The UML sequence diagram in Figure 7 shows a possible scenario where the protection of a message is verified using Ws-Trust, XML Digital Signature and XML Encryption to validate tokens, and to check and cipher/decipher messages SOAP respectively. It retrieves security policy of the security policy service; it validates the token obtained using WS-Trust of the Security Token Manager (calling to the validate operation of its interface); it checks the SOAP message and when the message has been correctly checked according to the security policy it will decipher the SOAP message obtaining the original message properly protected.

This pattern protects the message (integrity, confidentiality and authentication), thereby if we have a service that sends and receives SOAP messages, we must be sure that messages are valid and do not produce threats for the application, that is, we can use this pattern for establishing a basic architecture for protecting and verifying the protection of SOAP messages.

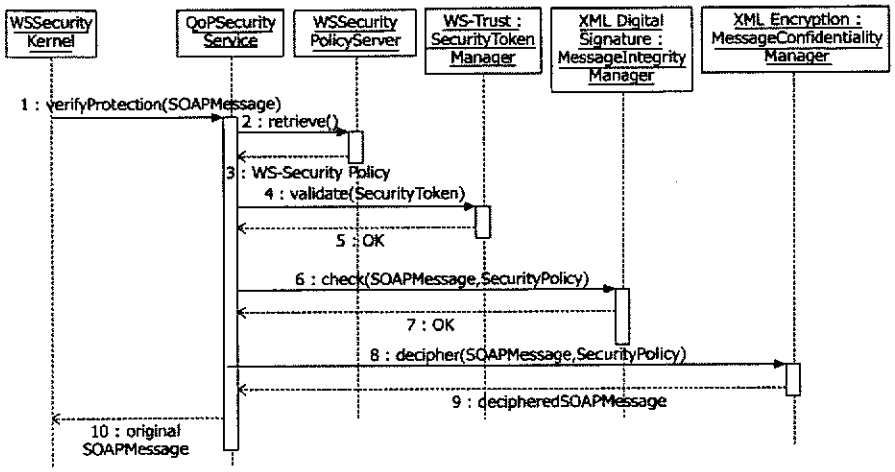


Figure 7.
UML sequence diagram of the "QoP" pattern: verifyProtection

Conclusions

Architects must make design decisions early in a project lifecycle. Many of them are difficult to validate and test until parts of the system are actually built. Due to the difficulty of validating early design decisions, architects rely on tried and tested approaches for solving certain classes of problems. This is one of the great values of architectural patterns. They enable architects to reduce risk by leveraging successful designs with known engineering attributes.

In the past, only software architects engaged in military application development had to learn complex security methodologies. The rapid expansion of e-commerce and internet applications has increased the need for an adequate application security for practically all enterprise applications. Therefore, software architects of enterprise applications are faced with a difficult choice.

This work has presented a possible relationship between security requirements, patterns and technologies that help us develop a security architecture using the most appropriate patterns that cover WS security requirements specific to the internet-based application, using WS standards for a correct implementation of the security services needed in our design and architecture.

We will take these security requirements specified by the process and then create a draft of the candidate security architecture. This activity qualifies the architectural decisions through a well-defined risk analysis and trade-off analysis processes (PWSSec) in order to identify the risks and trade-offs and how to mitigate them. This candidate architecture will also identify a set of security patterns that cover all of the security requirements within the component architecture and we will detail them in a high-level way, addressing the known risks, exposures, and vulnerabilities.

Our future work will be that of studying the different security patterns and obtaining a method with which we can find out what pattern is the best, for every requirement type selected, of the possible patterns to use according to a specific security property (performance, reliability, degree security, flexibility, etc) and then we can use them in our reference architecture previously created.

References

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I. and Angel, S. (1977), *A Pattern Language: Towns, Buildings, Construction*, Oxford University Press, New York, NY.

Anderson, S., Bohren, J., Boubez, T., Chanliau, M., Della-Libera, G., Dixon, B., Garg, P. and Gudgin, M. *et al.* (2005), *Web Services Trust Language (WS-Trust)*, February, available ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf

Ashley, P., Hada, S., Karjoth, G. and Schunter, M. (2002), "E-P3P privacy policies and privacy authorization", Workshop on Privacy in the Electronic Society, WPES'02, Washington, DC.

Bass, L., Clements, P. and Kazman, R. (2003), *Software Architecture in Practice*, Addison-Wesley, Reading, MA.

Berry, C.A., Carnell, J., Juric, M.B., Kunnumpurath, M.M., Nashi, N. and Romanosky, S. (2002), Chapter 5: Patterns Applied to Manage Security, J2EE Design Patterns Applied.

Bilorusets, R., Bosworth, A., Box, D., Cabrera, L.F., Collison, D., Ferguson, D., Ferris, C. and Freund, T. (2004), *Web Services Reliable Messaging Protocol (WS-ReliableMessaging)*, March, available at:ftp://www6.software.ibm.com/software/developer/library/ws-reliablemessaging200403.pdf

Buschmann, F. (1999), "Building software with patterns", paper presented at the 2nd European Conference on Pattern Languages of Programs (EuroPLoP 1999), Irsee, Germany.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M. (1996), *Pattern-Oriented Software Architecture: A System of Patterns*, John Wiley, New York, NY.

CERT (2006), *Statistics 1988-2006*, available at: www.cert.org/stats/cert_stats.html#incidents

Cranor, L., Langheinrich, M. and Marchiori, M. (2002), *A P3P Preference Exchange Language 1.0 (APPEL1.0). W3C Working Draft*, available at: www.w3.org/TR/2002/WD-P3P-preferences-20020415/

Curbera, F., Nagy, W.A. and Weerawarana, S. (2001), "Web services: why and how", Workshop on Object Orientation and Web Services OOWS2001, Tampa, FL.

Cheng, B.H.C., Konrad, S., Campbell, L.A. and Wassermann, R. (2003), "Using security patterns to model and analyze security requirements", High Assurance Systems Workshop (RHAS 03) as part of the IEEE Joint International Conference on Requirements Engineering (RE 03), Monterey Bay, CA.

Das Neves, F. and Garrido, A. (1998), *BodyGuard, Pattern Languages of Programs III*, Addison-Wesley, Reading, MA.

Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Bell, S. and Ylonen, T. (1999), SPKI Certificate Theory, Internet Eng. Task Force RFC 2693.

Evans, C., Chappell, D., Bunting, D., Tharakan, G., Shimamura, H., Durand, J., Mischkinsky, J. and Nihei, K. *et al.* (2003), Web Services Reliability (WS-Reliability) Ver1.0. January 8, available at: www.oracle.com/technology/tech/webservices/htdocs/spec/WS-ReliabilityV1.0.pdf

Fernandez, E.B. (2002), "Patterns for operating systems access control", paper presented at the 9th Conference on Pattern Languages of Programs, PLoP 2002, Allerton Park, IL.

Fernandez, E.B. and Pan, R. (2001), "A pattern language for security models", paper presented at the 8th Conference on Pattern Languages of Programs, PLoP 2001, Allerton Park, Monticello, IL.

Fernandez, E.B., Petrie, M.L., Seliya, N. and Herzberg, A. (2003), "A pattern language for firewalls", paper presented at the 10th Conference on Pattern Languages of Programs (PLoP'2003), Allerton Park, Monticello, IL.

Firesmith, D.G. (2003), "Engineering security requirements", *Journal of Object Technology*, Vol. 2 No. 1, pp. 53-68.

Firesmith, D.G. (2004), "Specifying reusable security requirements", *Journal of Object Technology*, Vol. 3, pp. 61-75.

Flanders, R. and Fernandez, E.B. (1999), "Data filter architecture pattern", paper presented at the 6th Conference on Pattern Languages of Programs, PLoP 1999, Allerton Park, Monticello, IL.

Ford, W., Hallam-Baker, P., Fox, B., Dillaway, B., LaMacchia, B., Epstein, J. and Lapp, J. (2001), *XML Key Management Specification (XKMS). W3C Note 30*, VeriSign Inc, Microsoft Corporation, webMethods Inc., London.

Fox, S. (2001), *Collection and Dissemination of Computer and Internet Security Related Information (Alerts, Advisories, Incident Notes, Vulnerability Notes, Summaries and other Bulletins)*, SANS.org, Washington, DC, August 21.

Gutiérrez, C., Fernández-Medina, E. and Piattini, M. (2004), "Web services security: is the problem solved?", *Information Systems Security*, Vol. 13, pp. 22-31.

Gutiérrez, C., Fernández-Medina, E. and Piattini, M. (2005a), "PWSSec: process for web services security", paper presented at the IEEE International Conference on Web Services, Orlando, FL.

Gutiérrez, C., Fernández-Medina, E. and Piattini, M. (2005b), "Web services-based security requirement elicitation", paper presented at 1st International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements (SOCCER 2005), in conjunction with RE 05 – 13th IEEE International Requirements Engineering Conference, Paris, France.

Gutiérrez, C., Fernández-Medina, E. and Piattini, M. (2005c), "Web services enterprise security architecture: a case study", Workshop on Security on Web Services, Fairfax, VA.

Housley, R., Ford, W., Polk, W. and Solo, D. (1999), Internet X.509 Public Key Infrastructure Certificate and CRL Profile, Internet Eng. Task Force RFC 2459, January.

IBM (2002), *Security in a Web Services World: A Proposed Architecture and Roadmap. White Paper from IBM Corporation and Microsoft Corporation, Version 1.0*, available at: www.verisign.com/wss/architectureRoadmap.pdf

IDC (2005), *Consumption of Web Services Will Greatly Increase through 2009*, available at: www.idc.com/getdoc.jsp?containerId=prUS00190705

Imamura, T. and Tatsubori, M. (2003), "Patterns for securing web services messaging", OOPSLA'03 Workshop for Web Services and Service Oriented Architecture Best Practice and Patterns, Anaheim, CA.

Kis, M. (2002), "Information Security antipatterns in software requirements engineering", paper presented at the 9th Conference on Pattern Languages of Programs (PLoP'2002), Allterton Park, Monticello, IL.

Lee Brown, J.F., DiVietri, J., Diaz de Villegas, G. and Fernandez, E.B. (1999), "The authenticator pattern", paper presented at the 6th Conference on Pattern Languages of Programs, PLoP 1999, Allerton Park, Monticello, IL.

Lehtoren, S. and Pärssinen, J. (2002), "Pattern language for cryptographic key management", paper presented at the 7th European Conference on Pattern Languages of Programs (EuroPlop'2002), Irsee, Germany.

OASIS (2003), *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*, available at: www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf

OASIS (2005), *Quality Model for Web Services*, Available at: www.oasis-open.org/committees/download.php/15910/WSQM-ver-2.0.doc.

OASIS (2006a), *Web Services Security. X.509 Certificate Token Profile 1.1. OASIS Standard Specification*, available at: www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf

OASIS (2006b), *Web Services Security. Username Token Profile 1.1. OASIS Standard Specification*, 1 February, available at: www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf

OASIS (2006c), *Web Services Security. Kerberos Token Profile 1.1. OASIS Standard Specification*, 1 February, available at: www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf

OASIS (2006d), *Web Services Security. SOAP Message Security 1.1 (WS-Security 2004). OASIS Standard Specification*, 1 February, available at: www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

Ramachandran, J. (2002), *Designing Security Architecture Solutions*, John Wiley, New York, NY.

Romanosky, S. (2001), *Security Design Patterns*, available at: www.cgisecurity.com/lib/securityDesignPatterns.html

Romanosky, S. (2002), "Enterprise security patterns", paper presented at the 7th European Conference on Pattern Languages of Programs (EuroPlop'02), Irsee, Germany.

Rosado, D.G., Gutiérrez, C., Fernandez-Medina, E. and Piattini, M. (2006), "A study of security architectural patterns", paper presented at the 1st International Conference on Availability, Reliability and Security (ARES 2006), Vienna, Austria, IEEE Computer Society, available at: http://csdl.computer.org/dl/proceedings/ares/2006/2567/00/25670358.pdf

Steel, C., Nagappan, R. and Lai, R. (2005), *Core Security Patterns*, Prentice-Hall, Englewood Cliffs, NJ.

Todd, S., Parr, F. and Conner, M. (2001), *A Primer for HTTPR. An Overview of the Reliable HTTP Protocol*, IBM, available at: www-128.ibm.com/developerworks/webservices/library/ws-phtt/

W3C (2004), *Services Architecture*, available at: www.w3.org/TR/2004/NOTE-ws-arch-20040211/

WS-I (2005), *Security Challenges, Threats and Countermeasures Version 1.0.T. Report*, available at: www.ws-i.org/Profiles/BasicSecurity/SecurityChallenges-1.0-20050507.doc

Yagüe, M.I. and Troya, J.M. (2002), "A semantic approach for access control in web services", paper presented at the Euroweb 2002 International Conference, W3C & British Computer Society Electronic Workshops in Computing (eWiC), Oxford.

Yagüe, M.I., Maña, A. and López, J. (2005), "A metadata-based access control model for web services", *Internet Research Journal: Electronic Networking Applications and Policy*, Vol. 25 No. 1, pp. 99-116.

Yoder, J. and Barcalow, J. (1997), "Architectural patterns for enabling application security", paper presented at the 4th Conference on Patterns Language of Programming, PLop 1997, Monticello, IL.

Corresponding author
David G. Rosado can be contacted at: David.GRosado@uclm.es

Author guidelines

Internet Research

Copyright
Articles submitted to the journal should be original contributions (see www.emeraldinsight.com/info/copyright/plagiarism_full.jsp for the journal plagiarism policy) and should not be under consideration for any other publication at the same time. Authors submitting articles for publication warrant that the work is not an infringement of any existing copyright and will indemnify the publisher against any breach of such warranty. For ease of dissemination and to ensure proper policing of use, papers and contributions become the legal copyright of the publisher unless otherwise agreed. Submissions should be sent to:

The Editor
Dr David G. Schwartz, Information Systems Division Head, Graduate School of Business Administration, Bar-Ilan University, Israel. Tel: +972 54 890 060; Fax: +972 3535 3182; E-mail: dschwar@mail.biu.ac.il
(Articles created in Microsoft or compatible format may be submitted by e-mail attachment. Authors are requested not to compress files.)

Editorial objectives
Internet Research, established in 1991, was the first scholarly journal to devote itself to collecting serious research about the internet and its applications and uses, specifically organizational.

The journal aims to discuss, assess and foster understanding of the role of wide-area multipurpose computer networks, specifically the internet.

Editorial scope
The primary focus is business and organizational applications of the Internet, such as marketing, promotion, data collection, research, customer service, publishing, educational, legal and security issues.

The Editor encourages a wide range of submissions and contributions, including research findings, critical essays, trend analyses, anecdotal experiences, product and service reviews and commentaries, book and other publication reviews, and other areas of interest.

Subscribers to the journal include executives, educators and researchers in commercial, academic, government and library organizations.

The reviewing process
Papers are normally reviewed by two or more members of the journal's review board. Where possible, submissions will be blind-reviewed using electronic network technology.

For this reason, all manuscripts should be accompanied by a copy of the article on disk or should also be submitted by e-mail.

Manuscript requirements
The manuscript should be submitted as a Word or rtf document, formatted with double line spacing with wide margins. All authors should be shown and **author's details** must be printed on a separate sheet and the author should not be identified anywhere else in the article.

As a guide, articles should be between 2,000 and 6,000 words in **length**. A **title** of not more than eight words should be provided. A brief **autobiographical note** should be supplied including full name, affiliation, e-mail address and full international contact details.

Authors must supply a **structured abstract** set out under 4-6 sub-headings: Purpose; Methodology/approach; Findings; Research limitations/implications (if applicable); Practical implications (if applicable); and the Originality/value of paper. Maximum is 250 words in total. In addition provide up to six **keywords** which encapsulate the principal topics of the paper and categorise your paper under one of these **classifications**: Research paper, Viewpoint, Technical paper, Conceptual paper, Case study, Literature review or General review. For more information and guidance on structured abstracts visit: www.emeraldinsight.com/structuredabstracts

Where there is a **methodology**, it should be clearly described under a separate heading. **Headings** must be short, clearly defined and not numbered. **Notes** or **Endnotes** should be used only if absolutely necessary and must be identified in the text by consecutive numbers, enclosed in square brackets and listed at the end of the article.

All **Figures** (charts, diagrams and line drawings) and **Plates** (photographic images) should be submitted in both electronic form and hard copy originals. Figures should be of clear quality, black and white and numbered consecutively with arabic numerals.

Electronic figures should be either copied and pasted or saved and imported from the origination software into a blank Microsoft Word document. Figures created in MS Powerpoint are also acceptable. Acceptable standard image formats are: .eps, .pdf, .ai and .wmf. If you are unable to supply graphics in these formats then please ensure they are .tif, .jpeg, .bmp, .pcx, .pic, .gif or .pct at a resolution of at least 300dpi and at least 10cm wide. To prepare **screenshots** simultaneously press the "Alt" and "Print screen" keys on the keyboard, open a blank Microsoft Word document and simultaneously press "Ctrl" and "V" to paste the image. (Capture all the contents/windows on the computer screen to paste into MS Word, by simultaneously pressing "Ctrl" and "Print screen".)

For photographic images (plates) good quality original photographs should be submitted. If supplied electronically they should be saved as .tif or .jpeg files at a resolution of at least 300dpi and at least 10cm wide. Digital camera settings should be set at the highest resolution/quality as possible.

In the text of the paper the preferred position of all figures and plates should be indicated by typing on a separate line the words "Take in Figure (No.)" or "Take in Plate (No.)". Supply succinct and clear captions for all figures and plates.

Tables should be typed and included as part of the manuscript. They should not be submitted as graphic elements.

References to other publications must be in Harvard style and carefully checked for completeness, accuracy and consistency. This is very important in an electronic environment because it enables your readers to exploit the Reference Linking facility on the database and link back to the works you have cited through CrossRef. You should include all author names and initials and give any journal title in full.

You should cite publications in the text: (Adams, 1997) using the first named author's name. At the end of the paper a reference list in alphabetical order should be supplied:

For books: surname, initials, (year), *title of book*, publisher, place of publication, e.g. Fallbright, A. and Khan, G. (2001), *Competing Strategies*, Outhouse Press, Rochester.

For book chapters: surname, initials, (year), "chapter title", editor's surname, initials, *title of book*, publisher, place of publication, pages, e.g. Bessley, M. and Wilson, P. (1999), "Marketing for the production manager", in Levicki, J. (Ed.), *Taking the Blinkers off Managers*, Broom Relm, London, pp. 29-33.

For journals: surname, initials, (year), "title of article", *journal name*, volume, number, pages, e.g. Greenwald, E. (2000), "Empowered to serve", *Management Decision*, Vol. 33 No. 5, pp. 6-10.

For electronic sources: if available online the full URL should be supplied at the end of the reference.

Final submission of the article
Once accepted for publication, the editor may request the final version as an attached file to an e-mail or to be supplied on a **CD-ROM** labelled with author name(s); title of article; journal title; file name.

Each article must be accompanied by a completed and signed **Journal Article Record Form** available from the Editor or on www.emeraldinsight.com/jarform

The manuscript will be considered to be the definitive version of the article. The author must ensure that it is complete, grammatically correct and without spelling or typographical errors.

The preferred file format is Word. Other acceptable formats for technical/maths content are Rich text format and TeX/LaTeX.

Technical assistance is available by contacting Mike Massey at Emerald. E-mail: mmassey@emeraldinsight.com

Authors' Charter

Your rights as a contributor to an Emerald journal

Emerald's copyright principles

Emerald* seeks to retain copyright of the articles it publishes, without the authors giving up their rights to republish or reproduce their articles on paper or electronically, subject to acknowledgment of first publication details.

Emerald's commitment to you

- An innovative publishing service which is timely, efficient, responsive and courteous
- Quality peer reviewed journals with editorial teams of distinction
- A named individual to keep you informed of publication progress
- Complimentary journal copy **plus** reprints of your paper
- An editorial and production policy which encourages accuracy and reduces submission to publication times
- On-line resources, forums and conferences to assist you with your research
- Responsible rights management to promote and safeguard the integrity of your work, encourage citation and wider dissemination
- Liberal reproduction rights and premium permissions service for yourself and subscribing organizations to serve the interests and needs of the scholarly community
- Additional benefits of Literati Club membership
- Consideration for nomination of the Annual Awards for Excellence to reward outstanding work
- Outstanding Doctoral Research Awards for our author community.

*Emerald - Electronic Management Research Library Database. Emerald is a trading name of MCB UP Ltd.

The full text of Emerald's Authors' Charter can be found at
www.emeraldinsight.com/charter

To discuss any aspect of this charter please contact us by e-mail at
literaticlub@emeraldinsight.com

Tel +44(0) 1274 777700 Fax +44 (0) 1274 785200
Literati Club, Emerald, 60/62 Toller Lane, Bradford BD8 9BY, United Kingdom.

Emerald