Web services security evaluation considerations

Elias Pimenidis*

School of Computing and Technology,
University of East London,
London E16 2RD, UK
E-mail: e.pimenidis@uel.ac.uk
*Corresponding author

Christos K. Georgiadis

Department of Applied Informatics,
University of Macedonia,
Thessaloniki GR 540 06, Greece
E-mail: geor@uom.gr

Abstract: Web services development is a key theme in utilisation of the commercial exploitation of the semantic web. Paramount to the development and offering of such services is the issue of security features and the way these are applied in instituting trust amongst participants and recipients of the service. Implementing such security features is a major challenge to developers as they need to balance these with performance and interoperability requirements. Being able to evaluate the level of security offered is a desirable feature for any prospective participant. The authors attempt to address the issues of security requirements and evaluation criteria, while they discuss the challenges of security implementation through a simple web service application case.

Keywords: web services; web services security evaluation; web services security implementation; web service user trust.

Reference to this paper should be made as follows: Pimenidis, E. and Georgiadis, C.K. (2009) 'Web services security evaluation considerations', *Int. J. Electronic Security and Digital Forensics*, Vol. 2, No. 3, pp.239–252.

Biographical notes: Elias Pimenidis is a Senior Lecturer at the University of East London in the UK and the programme leader for the MSc Information Security and Computer Forensics. The core of his research work focuses on e-business and e-government development projects. His interest in security issues is on online transactions and in particular, on the effect, these could have on e-government implementations. Other research interests include the evaluation of web services, knowledge management systems and the use of computer games for educational purposes. He is also a visiting Lecturer at the Informatics laboratory of the Agricultural University of Athens, Greece.

Christos K. Georgiadis received his BSc in Mathematics and his PhD in Information Systems Security – Access Control for Web Databases from the Aristotle University of Thessaloniki, Greece (1987). His research interests include the areas of e-commerce and m-commerce technologies, e-security and web services supporting technologies. Research productivity is summarised in various articles in cooperation with other researchers, in international journals and international conferences proceedings. He has participated in several research projects in Greece and in European Union related to information systems security and e-health. He is a professional member of ACM, and the SIGEcom. He has worked as a Research Associate in the Aristotle University of Thessaloniki, and as a part-time Lecturer in the University of Thessaly (Greece). Since 2004, he works as a Lecturer in the University of Macedonia – Department of Applied Informatics (Thessaloniki, Greece).

1 Introduction

A web service is a software system identified by a URL, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the web service in a manner prescribed by its definition, using XML-based messages conveyed by internet protocols. This definition has been published by the world-wide-web consortium W3C, in the web services architecture document (Booth et al., 2004).

The web service model consists of three entities, the service provider, the service registry and the service consumer. The key requirements for any service provider are: interoperability, security and performance. Most researchers focus on a common belief is that interoperability of web service must come along with considerable performance penalty (Georgiadis and Pimenidis, 2007). One common finding is that all three could be affected by the automatic choice of partners in forming a web service and that all three could mutually affect each other (Casola et al., 2007).

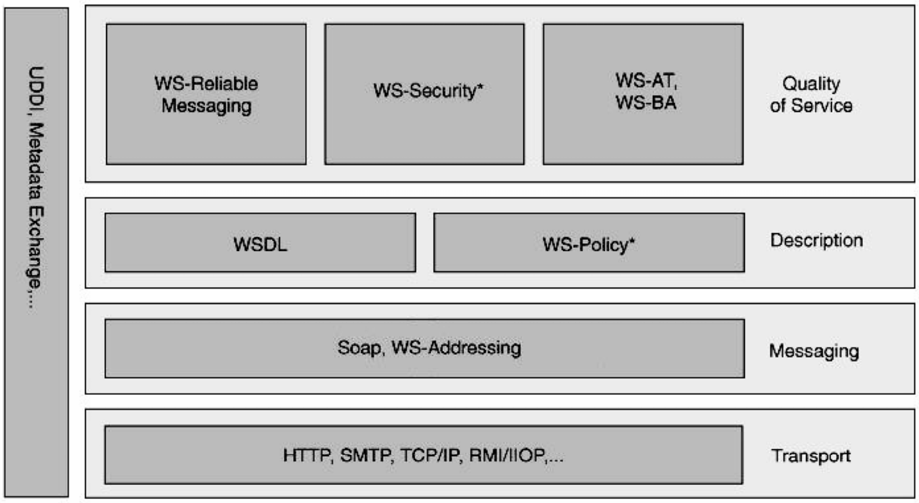
The most attractive feature of web service is its interoperability in a heterogeneous environment, and exposing existing applications as a web service increases their reach to different client types. Security measures are not something that can be added in a certain system's architecture, without having thought of them and design them at the very early stages (Chen et al., 2007). The integration of context into web service composition/transaction ensures that the requirements and constraints on these web service (either security- or interoperability-oriented) are taken into account. Context may support web service in their decision-making process when it comes to whether accepting or rejecting participation in a transaction (Casola et al., 2007; Georgiadis and Pimenidis, 2007).

2 Web services and security

2.1 Architecture layers and relative specifications

The high-level architecture of systems exploiting web service technology is essentially a stack of service-oriented capabilities. The bottom layers (namely transport and messaging layers) present its capabilities to cope with various transport protocols to communicate between a service and a requester, as well as to deal with messages. Major messaging specifications are XML (it provides the interoperable format to describe message content between web service), SOAP (it defines an extensible enveloping mechanism) and web service-addressing (it provides an interoperable way of identifying message senders and receivers; Figure 1).

Figure 1 Web service architecture



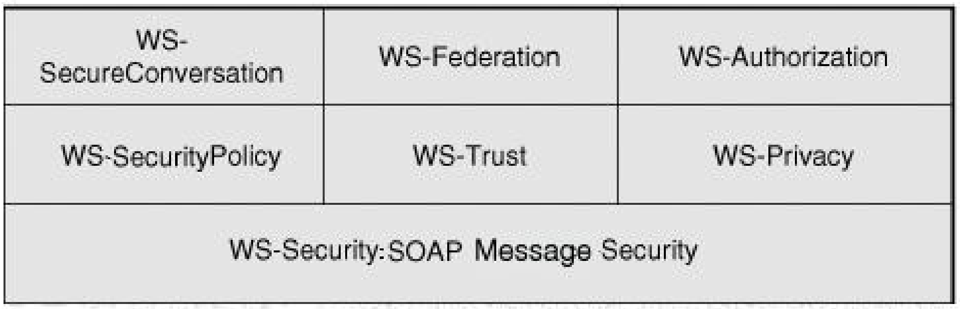
Source: Weerawarana et al. (2006).

The next layer, the description layer, deals with the description of services in terms of binding mechanisms and functions supported, as well as the quality of services of these functions. WSDL is an XML format for describing network services. It uses metadata to provide a set of endpoints that operate on messages containing either document-oriented or procedure-oriented information. Although WSDL describes what a service can do by providing a definition of the business interface (including business operations such as debit/credit/transfer), it does not provide information about how the service delivers its interface or what the service expects of the caller when it uses the service. The web service-policy specifications deal with this kind of issues and provide an extensible framework for web service constraints and conditions that allows a uniform expression of the available options. Thus, when multiple choices are possible, it is capable to provide support for determining valid intersections of conditions and constraints. Moreover, web service consumers and providers are not confused with multiple domain-specific mechanisms, because policy specifications enable constraints and conditions associated with various domains (e.g. security, transactions, etc.) to be composeable.

These issues are certainly of critical importance regarding security concerns, and so web service-policy specifications are actually a key security component of the overall architecture. To be exact, the actual quality of services (in terms of supporting transactions, reliable messaging and security) resides in a separate layer, and is based on appropriate parameterisation via policies. Focusing in security-oriented capabilities, we have first to mention the web service-reliable messaging specification, which may ensure any combination of the following assurances: the messages are delivered in the same order in which they were sent, no duplicate messages are delivered and each message that is sent is delivered at least one time.

Making web service interactions reliable in the presence of failures is certainly an important issue, but making web service-reliable even when the network, the web service itself or both are under possible security attacks requires a specific family of specifications, namely the web service-security specifications.

Figure 2 The web service-security family of specifications



WS-SecureConversation	WS-Federation	WS-Authorization
WS-SecurityPolicy	WS-Trust	WS-Privacy
WS-Security:SOAP Message Security		

Source: Weerawarana et al. (2006).

2.2 Need for web service security-related specifications

The web service security-related specifications define the required policies to properly secure the web service interactions. Their task is to set the constraints and capabilities of a web service, and actually they do not intend to substitute any existing security technologies. On the contrary, web service security-related specifications enlarge and merge existing security infrastructures and concretely define how these can be used in an interoperable way.

Multi-party and/or multi-hop web service interactions cannot be secure without web service security-oriented specifications. Existing technologies, such as SSL/TLS, IPSec and HTTP-S, have no end-to-end security or persistency. They are just capable to provide in-transit confidentiality and integrity, securing point-to-point connections (e.g. between the end user and company's systems in B2C transactions or between companies' systems in B2B transactions). But, these attributes are lost after the message is delivered, and this is surely a problem considering that in web service interactions there are always intermediaries placing significant communication issues: although they should not have access in general to sensitive data (because they are not always completely trusted), they might need to inspect or even alter at least some parts of passing messages (Figure 2).

Moreover, the variance of intermediaries requests extremely flexibility from web service-security means, to accommodate many different security models. Web service integrate multiple systems with different security domains and technology, and so there is a need for a mechanism to exchange or translate security metadata from one domain to another (e.g. the end user is authenticated by a certain company's system, and this authentication information is then propagated trustworthy and meaningfully to other companies).

2.3 Web service-security specifications

A set of fundamental and conceptual web service security-oriented standards, namely the web service-security family of specifications, collectively form a web service-centric security framework that facilitates the message-level security required to end-to-end protection (Erl, 2004). This web service-security framework acknowledges the fact that web service technology is often used to integrate several existing applications available on the network. These applications might be extremely diverse in terms of hardware, operating systems, middleware and in their configurations. In addition, they might reside in different security domains that have different security policies and are dependent on diverse security infrastructure, such as PKI and Kerberos (Weerawarana et al., 2006).

To solve the problem of translating security information from one system in one security domain to another system in another security domain, the web service-security framework defines standard syntax and semantics to express the security information, and rules to translate it. Specific terminology is used to describe this type of concerns. When a service requester makes a request of a service to a provider, it asserts a claim regarding its security clearness. It is then up to the service provider to validate this claim. A service requester may provide a number of claims to communicate different aspects of its security status. This set of claims is contained within a security token. Security tokens can be signed with a signing authority, allowing them to be further verified by the recipient of the message containing the token (Erl, 2004). Some tokens might be issued by a trusted third-party. Web service-security defines a standard XML format for transporting security tokens. For example, an X.509 certificate is usually issued by a certification authority. In the web service-security model, a trusted third-party is called a security token service because one of its tasks is to issue security tokens. The security token service might be well-known in a public network, such as an internet certification authority, or it might be an infrastructure service within an enterprise's network.

Although web service-security framework as it is defined today does not solve all issues, it is certainly a firm step in that direction, thanks to its flexibility and extensibility. Web service-security helps to solve these problems by defining the following (Weerawarana et al., 2006).

- A standard interoperable format (schema) for transporting security information (identity tokens, signature elements, and encryption elements and claims).
- A set of concrete security policy documents (extensions of web service-policy) that allow sites and web service to document their support for web service-security and their requirements on callers. For example, a site might use web service-policy to document which messages (or parts of messages) must be signed and which certificate authorities the site may use.
- A standard web service interface that web service can use to create, exchange and validate security tokens issued by other domains, and that requesters can use to translate credentials from one domain into tokens accepted by another domain.

Thus, the basic aspects of web service-security layers are the following (Chatterjee and Webber, 2004; Weerawarana et al., 2006).

1 *Web service-security. SOAP message security*. It is built on the SOAP specification and specifies how to sign and secure SOAP messages.

2 *Web service-securitypolicy*. It specifies a generic format through which to describe the security capabilities and requirements for SOAP message senders and receivers (both intermediaries and endpoint consumers). Existing corporate security policies can be expressed through policy assertions that can subsequently be applied to groups of services.

3 *Web service-trust*. It specifies and describes the model for establishing and coordinating trust relationships between multiple parties.

Web service-trust specification is very important, as it actually defines a conceptual model for using a web service. A requester examines the web service-securitypolicy statements associated with a web service. The policy statements specify the type and authority of security tokens that the web service requires for messages it processes. If the requester does not possess such acceptable tokens, it must obtain them. One way is to contact a security token server that the web service's policy statements identify as acceptable (Weerawarana et al., 2006).

Web service-trust also defines a model for challenges. Upon receiving a message with a security token in it, a service might send a challenge, forcing the requester to demonstrate its right to use the token. If the requester passes an X509 certificate, the service might respond with a challenge that contains some random data. The requester must sign the random data with the private key associated with the certificate, demonstrating proof of possession. The challenge model strengthens security by eliminating some attacks.

Web service-trust provides support for specifying key sizes and algorithms in the request and response messages. The specification also provides support for passing policy information in messages. Each web service endpoint logically implements a trust engine that must understand the web service-security and web service-trust model. The trust engine of a web service must verify the following.

- The security token is sufficient to comply with the web service's policy, and the message conforms to the policy (for example, the necessary elements are encrypted or signed).
- The security tokens are proven signatures.
- The issuers of the security tokens (including all related and ancestral security tokens) are trusted by this site to issue the claims they have made.

The trust engine might need to send tokens to a security token service to exchange them for other security tokens, which it can use directly in its evaluation. If the trust engine determines that the conditions are met and the requester is authorised to perform the operation, the web service can process the web service request within the aforementioned trust model.

The remaining layers of web service-security family are the following (Chatterjee and Webber, 2004; Weerawarana et al., 2006).

1 *Web service-secureconversation*. It is built on base web service-security and web service-trust to specify how web service can manually manage and authenticate security contexts. It includes describing how service requesters can authenticate web service as well as how web service can authenticate messages from service requesters.

2 *Web service-federation*. It is built on all previous specifications to specify how to broker and manage heterogeneous, federated trust contexts. Web service-federation itself describes how to implement a federation in a web service world. In particular, it focuses on the relationships between parties, and the high-level architecture that

supports these relationships. Web service-federation describes how to use the existing web service-security building blocks to provide federation functionality, including trust, single sign-on (and single sign-off), and attribute management across a federation.

3 *Web service-privacy*. It is built on base web service-security, web service-securitypolicy and web service-trust to specify a model by which organisations using web service can indicate preferences as well as conformance to particular privacy policies.

4 *Web service-authorisation*. It specifies how access policies for web service are specified and managed using a flexible and extensible authorisation language and format.

2.4 Web service-security standards: a work in progress

The solutions to the security challenges are still evolving and so pre-standard workarounds to security problems provide a critical aspect of the whole process. Some standards (e.g. web service-security, web service-securitypolicy) are mature and can be immediately employed. Others are still in development. The advantage of this situation lies on the modular character of the web service-* family of security standards: the implementations of current standards, the structure of the SOAP message body and the structure of the header portions of the messages related to the standards already in use, will not be impacted by the later adoption of newly matured standards (Brown, 2007). Thus, wherever these industry standards are applicable, appropriate and mature, they certainly should be adopted.

Temporary working solutions for the design problems that these emerging standards will eventually address, must be found in the meantime. Undoubtedly, this is the real challenge: to address the functional areas for which the standards are not yet full-grown. In most cases, web service-security standards require some supporting infrastructure and consequently an infrastructure investment in order to employ these standards is essential.

One of the main areas where web service-security framework needs supplementary methods and concepts is related with federating multiple security domains. The term federated identity has various meanings. To an individual user, it means the ability to associate his various application and system identities with one another. To an enterprise, federated identity provides a standardised means for directly providing services for trusted third-party users, or those that the business does not manage directly. An enterprise associates with others in a federation, such that the identities from one enterprise domain (or identity provider) are granted access to the services of the other enterprises (or service providers). Federated identity management refers to the set of business agreements, technical agreements and policies that enable companies to become partners. This lowers their overall identity management costs, improves the user experience and mitigates security risks in web service-based interactions.

Web service-federation defines mechanisms for brokering and federating trust, identity and claims. Federation is the overall term for a set of distinct, heterogeneous enterprises that want to provide an easy-to-use, single sign-on identity model to their users. Single sign-on means that after a user signs-on with one member of the federation, he can interact with other members without re-authentication. Enterprises can be corporate entities, internet service providers or associations of individuals. A federated environment differs from a traditional single sign-on environment in that there are no established rules limiting how enterprises transfer information about a user. However, there might be an established business policy for an enterprise's participation in the federation, much like there is today when companies decide to do business together.

Several limitations of the existing web service-federation specification have been identified. These limitations involve methods for establishing trust relationships, brokering and federating trust, publishing security policies, enforcing security requirements and exchanging security tokens. Considering the large number of participants interacting in a fully distributed web service system, it is impractical for entities to establish and to maintain explicit trust relationships with every customer and partner. While the web service-security framework describes how to specify trust policies, how to communicate trust and how to broker trust directly and indirectly through third-parties, it leaves the method for originating a trust policy to the organisation or the web service administrator. While it is constructive for the specifications to remain implementation independent, entities still need a basis for initially defining and subsequently maintaining their trust policies. If a service potentially acquires a large number of clients in a complex distributed system, then a web service designer or policy administrator must identify method(s) of practically enumerating all possible trust relationships with the different clients or their respective organisations (Van Dyke, 2004). Without having a process to establish trust relationships dynamically, it is difficult for entities to fully benefit from the loosely coupled, distributed nature of large-scale web service networks (WSN).

Web service-federation does not specify how security token exchanges can occur, given potential differences in the various domains' methods for authenticating users and services. Moreover, the specification does not suggest strategies for establishing trust relationships between entities, for mapping differences in the trustworthiness of security tokens across domains, and designating authorities to maintain and to publish security definitions and requirements (Van Dyke, 2004). Current research suggests advanced approaches for designing dynamic web service trust networks that confront these issues. Specifically, it discusses how additional primary concepts (such as trust levels, trust groups and trust authorities) can enhance existing web service-* security specifications. Trust levels can be thought of as mutually recognised, standardised identifiers of the 'level of trustworthiness' for a given user or service. For example, an entity that interacts with a user who holds an authentication ticket signed by one of its trusted partners should be able to trust this user to some degree, as outlined by the entity's trust policy. With these additions, systems can define dynamic trust policies and generate dynamic trust relationships. It is worth mentioning that proponents of web service believe that new web service technologies will expand federation as it exists for ATM networks into a generalised architecture that is applicable for many types of internet communications.

3 Platforms for security solutions

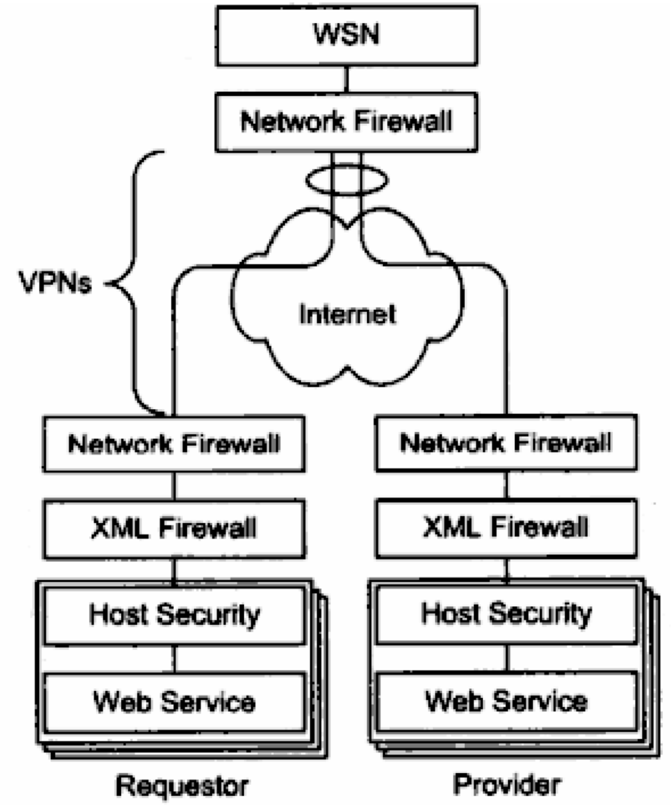
Web service-security specifications do not make network-layer or host-based security mechanisms unnecessary within a service-oriented communications framework, but only limit their role. A complete security solution should include multiple platforms. We will follow the path of a web service request message in the sample configuration of Figure 3, to illustrate an indicative way to combine the responsibilities of the various security platforms-solutions (Kaye, 2003).

- Data of critical importance, with high-level requirements for confidentiality and integrity, must be encrypted using host-based (or application-based) security mechanisms. By performing encryption (at requestor's side) and decryption (at provider's side) as close to the application as possible, data integrity and confidentiality are protected over the greatest percentage of its route. This platform is capable to perform more granular authentication and authorisation than is possible in the XML/application firewall.
- The XML/application firewall verifies XML syntax and checks documents against business rules. Moreover, it authenticates and verifies the authorisations of entities submitting external requests. Finally, it may encrypt data of less importance. Because it is separate from the web service, the XML firewall is able to provide security for multiple web service applications. By handling all encryption and decryption tasks, it provides a complete set of integrity and confidentiality services covering both component-level issues and end-to-end issues for multi-hop systems. It may perform authentication tasks at multiple levels, including multi-party and bi-directional authentications. Since it may read the content of web service messages, and to authenticate their authors, XML firewall is the ultimate authorisation mechanism: it may support loosely coupled authorisation models and it may include sophisticated rules-based engines to express complex authorisation logic.
- Between the network firewalls of the web service endpoints and the WSN, virtual private networks (VPNs) are implemented which have the primary responsibility for the integrity and confidentiality of data as it passes through the internet. Network firewalls and their network-layer security offer simple and effective transport-layer encryption for either temporary (using Secure Sockets Layer (SSL) protocol), or persistent (using VPN mechanisms) point-to-point connections. To establish a VPN, considerable business and technical negotiations between the parties are required. But even then, only a part of the web service-security challenges are solved: simple point-to-point links are used only by web service without intermediaries. Thus, network firewalls and VPNs can only play supporting roles for integrity, confidentiality and authentication. They cannot provide end-to-end related security attributes in a multi-hop architecture. Moreover, network firewalls are intended to prevent access and to hide systems, whereas web service requires the exposition to the outside world of those very same systems. In web service, as the web service architecture layers indicate, the general goal is to move security out of the lower network and transport-layers and into the upper message-oriented layers. This allows security concepts to be implemented independently of any particular network or transport protocol. Network- and transport-independent security is required for any message that will be routed over more than one protocol on the way to its final destination.
- The WSN offers security qualities as centralised and shared third-party services. It provides the transformation services that make one endpoint compatible with another. Examples of such mapping services are the selectively decryption of

received data, its re-encryption and re-transmission to the destination endpoint. Other services are logging data for non-repudiation purposes and authentication/authorisation functions if the two endpoints use different models and therefore require mediation. This is essentially the platform in which the sophisticated web service-security specifications of the previous paragraph are implemented.

- One alternative to deploying security solutions is using peripheral systems which are not part of the web service message path. Peripheral-service security solutions provide mainly authentication and authorisation as services (even as web service). Characteristic examples are: centralised identity management systems, sharing schemes of identities among business partners and general purpose authentication engines that support complex rules expressed in standardised XML.

Figure 3 Distributing the responsibilities of security platforms



Source: Kaye (2003).

4 Implementing a web service on an Apache server

A simple web service implementation project was undertaken to assess the suitability of development tools and the level of security that can be achieved with standard development environments and techniques. This case is only discussed here from the point of view of security features implemented and how the tools used supported the requirements for web service-security.

4.1 Development tools utilised

In deciding on the choice of tools, Open Source software was compared and contrasted to those offered by major brand tools – primarily Microsoft products. The final choice was the Linux operating system with Apache web server, PHP 5 server side script language and MySQL database.

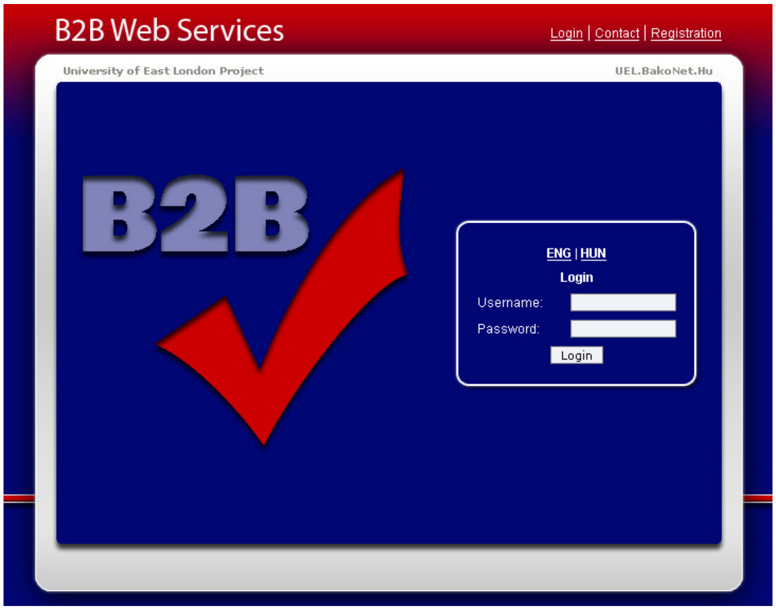
The main reasons for such a choice are as follows.

- Open source programs usually come with wide community support. Linux has an established reputation for high reliability and low hardware requirement and installation is quick and trouble free.
- Apache web server is one of the oldest and most commonly used server side development environments. It is easy and simple to configure and very stable. It also supports SSL which is of prime importance to web service applications.
- PHP has legendary compatibility with a huge range of operation systems. It also has a lot of additional modules. Version 5 of PHP offers high-level of support for object orientation, which is very useful in building applications in modular mode, as web services often need to expand and diversify.
- Finally, the selected database is MySQL. The choice of the newest version of MySQL which has almost the same features as the PostgreSQL was based on the fact that it is faster when handling simple queries.

4.2 Security features at registration

The registration is fast and simple with signed mandatory fields to identify those user's details that the user must give for the registration. These are standard details such as the username and the e-mail address. The given username is checked by the system to make sure that it constitutes a unique identifier for the user, so that each registered user has his or her own, individual identification in the database. Following this simple validation stage, the system adds the new user's details to the database and generates a random password. This password is sent to the registered user's e-mail address, and then the encrypted version of this password is stored in the database. For security reasons, pure text passwords are not stored anywhere. In this case, it is not possible to recover the original password because the used encryption method is a one way routine (common md5 encryption is used). The registered user will not have permission to use the system until the administrator changes his or her status from the default passive state to active (Figure 4).

Figure 4 Registration page with choice of language (see online version for colours)



The user level is identified by a flag system in the database. After the registration process is finished, the new user will be assigned the lowest user level status, signed with an 'R' – flag (registered). On this level, the users have access to the web service and can login to the user interface to get information about the current services and handle personal details.

The second level users can be assigned to is 'A' – flag (approved). It means that they have been granted privileges to add or modify the services (the user must know the MIGs of the input and the output to add or modify a service) and also all the options as the users on the lower level.

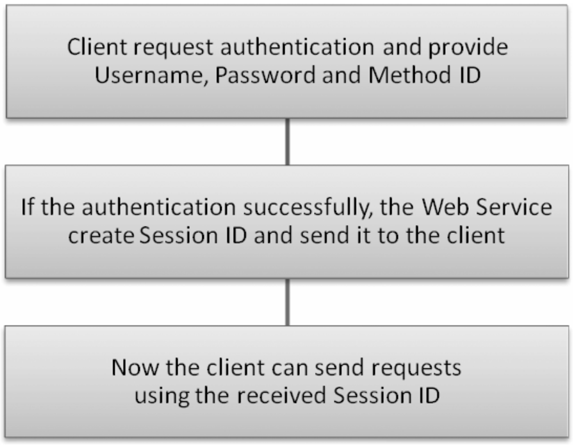
The highest level is identified by an 'S' – flag (site administrator), it means these users have permission to all lower level features and also user administration privileges.

4.3 Transaction security – session authentication

The above user policy system needs a strong authentication module to achieve the main objective; therefore, a number of controls are used during the login and for the period of the session. A separate database table is used to log the session details. At login, the user's internet protocol address and the user agent are checked. A time limit is also assigned for the user login and automatically finishes the session when it expires caused by inactivity. During the time of active connections, the time limit renews itself.

To use the web service requires authentication, which is sent at the beginning of the session. As soon as the verification is complete, the service sends a session ID to the client. This password is valid for the started session and identifies the user. The client sends back this authentication information to the server at all times when it is sending a request to the server. The system is also protected from SQL injection (Figure 5).

Figure 5 Transaction authentication schema


Client request authentication and provide Username, Password and Method ID
If the authentication successfully, the Web Service create Session ID and send it to the client
Now the client can send requests using the received Session ID

Since messaging itself does not provide secure transmission protocol, it brings high risks to both sides of the message exchange. Although traditional security technologies such as SSL and HTTPS can partially resolve this problem by encrypting messages transferred between two points, these point-to-point security technologies cannot ensure end-to-end security along the entire path from client to a web service in a complicated multi-tiered distributed system (Tang et al., 2006).

The web service has its own database to store translation methods, which can be used once the request reaches the service.

The service receives the request and identifies the 'method ID'. This ID comes by the first transaction, at the same time as the authentication information sent to the service. The system uses XML language to send these data and authenticate the session.

5 Conclusions

Web services are the current trend in establishing diverse environments for offering a choice to recipients and a plethora of potential customers to participants. Amongst the key desirable features of web services, security is the most critical and most sensitively balanced one: it must not be compromised, but at the same time must be implemented in such a way that it does not degrade the performance and undermine the interoperability of the service. These three properties of a web service are central to achieving the two major objectives of web service providers, to establish and maintain trust of recipients and to reach high volumes of users.

A variety of frameworks for implementation and evaluation of the level of security offered exist. The key in addressing the security requirements of a web service is in the trade offs in terms of performance and complexity of recipient participation. The simple implementation case discussed here demonstrates that commonly available tools can be utilised in web service implementation projects with good results in meeting desired security requirements. The honours are on the service orchestrator to ensure that the above three requirements are evenly balanced without any compromises in security.

References

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. (2004) 'Web services architecture', *W3C Working Group Note 11, February, W3C Technical Reports and Publications*. Available at: http://www.w3.org/TR/ws-arch/.

Brown, P. (2007) *Succeeding with SOA: Realizing Business Value through Total Architecture*. Boston, MA: Addison-Wesley.

Casola, V., Fasolino, A.R., Mazzocca, N. and Tramontana, P. (2007) 'A policy-based evaluation framework for quality and security in service oriented architectures', Paper presented in the Proceedings of the *IEEE International Conferences on Web Services ICWS 2007*.

Chatterjee, S. and Webber, J. (2004) *Developing Enterprise Web Services: An Architect's Guide*. NJ: Prentice Hall PTR.

Chen, S., Zic, J., Tang, K. and Levy, D. (2007) 'Performance evaluation and modeling of web services security', Paper presented in the Proceedings of the *IEEE International Conference on Web Services ICWS 2007*.

Erl, T. (2004) *Service-Oriented Architecture*. Upper Saddle River, NJ: Prentice Hall PTR.

Georgiadis, C.K. and Pimenidis, E. (2007) 'Proposing an evaluation framework for B2B web services-based transactions', Paper presented in the Proceedings of the *E-Activity and Leading Technologies Conference*, IASK, Porto, Portugal.

Kaye, D. (2003) *Loosely Coupled: The Missing Pieces of Web Services*. Marin County, California: RDS Press.

Tang, K., Chen, S., Levy, D., Zic, J. and Yan, B. (2006) 'A performance evaluation of web services security', Paper presented in the Proceedings of the *10th IEEE International Enterprise Distributed Object Computing Conference EDOC'06*.

Van Dyke, J.W. (2004) 'Establishing federated trust networks among web services', *BSc Thesis*, University of Virginia.

Weerawarana, S., Curbera, F., Leymann, F., Storey, T. and Ferguson, D. (2006) *Web Services Platform Architecture*. Upper Saddle River, NJ: Prentice Hall.