SEMANTICS-AWARE PERIMETER PROTECTION

Marco Cremonini, Ernesto Damiani and Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione
Università di Milano
26013 Crema - Italy
{cremonini,damiani,samarati}@dti.unimi.it

Abstract Web services security is becoming a critical concern for any organization adopting the XML-based Web services approach to application integration. While many access control techniques for Web services are becoming available, several issues still need to be solved in order to correctly split the burden of securing Web services between the perimetral and the service level. In this paper, a technique is presented able to make perimetral defences semantics-aware. Application-level *semantics-aware firewalls* enforce filtering rules directly on SOAP messages based on the nature of the services they request. Our semantics-aware firewalls rules are written using a flexible XML-based syntax that allows sharing metadata concepts with service level access control policies, supporting complex security policies that integrate perimetral defences with access control. Moreover, they can be quickly integrated into organizations' existing infrastructure, deployed rapidly and scaled as needed. Also, they integrate easily with existing infrastructure and can be operated by current staff, potentially achieving a low total cost of ownership with respect to service level solutions.

1. Introduction

It is widely acknowledged that open standards such as XML [3] and HTTP [9] offer reduced integration cost, boosting the potential for increasing organizations' capability to compete and succeed in today's global e-market. A well established trend toward a Web-centric scenario for both user-to-application and application-to-application interactions exploits Web services[1] flexibility, expressivity and standardization. As

[1]The term *Web services* describes Web application components build upon existing and emerging technology standards including XML, SOAP, WSDL and HTTP.

far as protocols are concerned, Web services growing success hints to a scenario where HTTP, with its secure variant HTTPS, is the only general-purpose application protocol used for a variety of services. The benefits of this evolution are clear:

- A simplified and homogeneous level 7 of the ISO/OSI stack with a single reference application protocol, i.e. HTTP, together with a standard remote method invocation technique, i.e. SOAP [2], overcomes the usual difficulties in integrating specialized services and proprietary platforms;

- a standard data interchange format, i.e. XML, makes integration of functions offered by heterogeneous systems much easier than it was in the past.

There is an additional claimed benefit of the Web-centric model, which has perhaps more to do with business than with technical reasons: tunnelling remote method invocations through HTTP connections, serves the goal of solving *firewall traversal* problems caused by the attitude of system administrators, and actually of most network security best practices, of blocking most application protocols at the corporate network perimeter. As a matter of fact, Web services firewall traversal capabilities introduce significant security risks, encouraging companies to selectively expose pieces of their enterprise applications. While a number of access control techniques for Web services are becoming available [7, 1], many researchers agree that the temptation to deal with Web services' security by access control alone should be resisted, because a number of security and operating issues are more effectively solved at perimetral level. In order to check SOAP traffic crossing organizations' perimeters, however, firewalls should become semantics-aware, at least to a degree. Commercial SOAP proxies are becoming available capable of performing sophisticated filtering of SOAP payload (to name but a few, Reactivity XML firewall, `http://www.reactivity.com`, and Quadrasis/Xtradyne SOAP Content Inspector, `http://www.quadrasis.com`, are examples of dedicated hardware appliances; Check Point Application Intelligence, `http://www.checkpoint.com`, instead, is the new SOAP proxy feature provided by the last release of market leader Firewall-1). However, the debate on their usage and impact is still at a preliminary stage: while enterprises have quickly come to realize the importance of securing XML, they are only beginning to understand the many operating issues and the corresponding costs involved in implementing XML security at the single service and at the perimetral level. In this paper, we introduce a new concept in application security, the

semantics-aware perimeter protection, aimed at cost-effectively sharing with service-level access control the heavy protection burden that Web services applications require, reducing the risks associated with integrating heterogeneous applications that cross organizational boundaries. In our approach, semantics-aware firewalls are not just simple filters acting on XML payload: rather, they systematically monitor XML-based messages between business partners, customers and others, enforcing rules written in a highly expressive, XML based language. When it is computationally feasible, rules enforced by semantics-aware firewalls may refer to service properties smoothly integrating with service level access control policies. Like our approach to XML and SOAP access control [5, 6, 7], our semantics-aware firewalls cleanly separate Web services security from the business logic in each application. This allows administrators to independently define and audit multiple-level security policies. The paper is structured as follows: Section 2 presents some basic concepts, relevant to our approach, referred to: Web services, the SOAP protocol, perimeter protection and network firewalls. Section 3 presents our new concepts of semantics-aware perimeter protection. It clarifies the context, which problems are tackled, how SOAP headers should be customized for perimetral security, and highlights some performance and manageability requirements. Section 4 describes how semantic-aware filtering rules are modelled in our framework, specifies their syntax and semantics, and presents a case-study example of semantic-aware policy. Section 5 draws our conclusions and sketches our plans for future works.

2. Basic concepts

2.1. Web services in a nutshell

Web services are the last evolution of basic Remote Procedure Calls (RPCs) and are aimed at providing a new form of distributed computing environment for business-to-business and business-to-consumer applications. Aside implementation differences with respect to RPCs, Web services emphasizes the adoption of standard technologies: a standard definition mechanism, standard lookup services and standard transport definitions. The goal is to make remote systems fully interoperating.

For the purpose of this paper, we are expecially interested in SOAP [2, 12], which is the key technology for carrying out Web services interactions on the net. In the following we give a brief overview of SOAP features that are relevant within the scope of this paper. In particular, we focus on the *SOAP HTTP Binding* scenario [12], which is represented by applications exchanging SOAP messages via HTTP.

SOAP is a stateless, one-way message exchange paradigm that, if combined with features of the underlying application protocol, could be used to realize different interaction patterns like: the *response* message exchange pattern, which is provided by the use of the HTTP *GET* method in a HTTP request to return a SOAP message in the body of the HTTP response; the *request/response* pattern, which is the one followed by using the HTTP *POST* method for passing SOAP messages in the body of HTTP requests and response messages; and the *request/multiple responses,* which is not directly supported by HTTP but should be implemented at application level [12]. We are interested in the *request/response* exchange message pattern, because expecially well-suited for RPC-like interaction, as Web services are.

The general structure of a SOAP message consists of an XML document composed by an `Envelope` element that contains two sub-elements: an `Header` and a `Body`. A SOAP `Header` contains auxiliary information such as directives or contextual information related to the processing of the message. SOAP is exensible in an application-specific manner by customizing such auxiliary header's information. Within a SOAP `Header`, sub-elements called `Header Blocks` represents data logically grouped that can be used by intermediaries handling a SOAP message between a sender and an ultimate receiver.

2.2. Perimeter protection concepts

The goal of perimeter protection is to deploy security measures at the perimeter of a corporate network. The perimeter of a corporate network is represented by the access points to and from external networks, like the network segments connecting the company to the Internet, external routers facing the Internet Service Provider's network, dial-in modem pools, Virtual Private Networks, or Wireless LANs.

The underlying assumption that motivates security measures based on the notion of corporate's perimeter is that relying on host-based security only is not an effective strategy. If interaction between untrusted domains and application hosts are not mediated by any filtering component, then *all* application servers are exposed to the whole range of security threats, either based on application related features or network accesses. Securing and hardening all application servers to be resistant to direct Internet attacks has been proven to be extremely hard, too complex and expensive to maintain for representing a general solution. Here comes perimetral defense that uncouples possibly hostile environments from data or resources to be protected and inspects interaction either at network or application level.

Firewalls are the main and most popular perimeter protection systems. A firewall is a system or a group of systems that enforces an access control policy on network traffic as it passes through access points [4].

Firewalls are typically classified into three main classes: *static packet filtering, stateful firewalls,* and *proxies.*

3. Semantics-aware perimeter protection

Traversing firewalls is more a management than a technological problem, since every firewall technology lets system administrators make the screening provided by firewalls completely transparent at will, with respect to every communication protocol. In other words, every remote method invocation protocol could traverse firewalls provided that system administrators configure the perimetral policy accordingly. Unfortunately, many applications and protocols have been proven to be highly insecure or simply too complex to be effectively secured. This situation has brought most system administrators to prevent some application-level traffic from flowing through perimeter defences. From the network security's viewpoint the fact that traversing firewalls is usually severely restricted should be regarded as an useful feature and the right attitude of a savvy administrator rather than a problem. ¿From the standpoint of application service providers, vendors or developers of Internet application services and corporate managers willing to exploit the business possibilities of the Internet, the constraints imposed by firewalls impair full exploitation of business opportunities. While one might be tempted to remove this impairment by entirely leaving application security to service-level access control policies, this choice is likely to prove unwise in the long run. Indeed, an inherent danger of the widespread adoption of Web services would be neglecting the value of perimeter protection by assuming that Internet applications and users will attempt to directly interact with Web services, with no intermediate screening and active monitoring. This would make the services' interfaces open to exposure: if compromised (and all components direcly exposed to Internet connections should be considered at high risk), application policies might be subverted and critical assets, databases and systems, might be put to risk.[2]

[2]Also, lowering the importance of perimetral defences policy would leave definition and management of corporate security to application managers alone, another choice that may prove organizationally unwise.

3.1. The firewall traversal problem

The SOAP protocol lets interactive application traffic to be tunnelled through regular HTTP communications, which makes the usual association between service and application port for TCP and UDP communications no longer descriptive of the type of service carried out by a certain flow of packets (i.e. all SOAP-based services appear directed to servers responding at port 80, the conventional HTTP daemon). The net effect of this use of HTTP as a general-purpose application protocol for distributed computing is that conventional firewalls, screening network traffic mostly at level 2 and 3 of the TCP/IP stack, are no longer able to efficiently filter out packets. Firewalling SOAP network traffic needs content-based inspection capabilities and detailed knowledge of Web services and their semantics.

With SOAP, there is no longer a straight separation between the perimetral and the service level with respect to data used for enforcing security policies. Both must use same data, either in the SOAP header or body, and with similar enforcing mechanisms. This makes perimetral and service levels more strictly coupled than in the past and calls for a new comprehensive and integrated approach to corporate security by developing systems that allow managing a uniform corporate security policy and permit to enforce it at both the perimeter and the service level.

3.2. Using SOAP headers for perimeter protection policies

SOAP headers, used to hold metadata associated with the requestor, have already been used to carry security credentials. A first proposal exploiting SOAP's headers for enforcing access control was presented by Damiani et al. [7]. The authors represented all the information characterizing the subject of a request by means of a custom header included in each SOAP call. This usage of SOAP's headers has then gained acceptance and has been adopted in popular frameworks like the Microsoft WS-Security specification [1], which proposes a set of SOAP extensions that can be used to implement integrity and confidentiality in Web services.

Both [1] and [7] are based on the assumption that a requestor of a service declares the possession of some personal credentials to the server providing the requested Web services, which, in turn, authenticate the requestor and provide the permissions corresponding to a defined security policy. This design choice is coherent with the overall design principle of SOAP interactions and security model, where requestor cre-

dentials, stored in the *Header* part, are firstly used to prune an access control policy, and then each service request is matched at run-time against the set of permissions available for the given requestor. However, it does not match two basic requirements of a perimeter protection policy, namely:

- *Filter Granularity.* There exists a trade-off between communication performances and complexity of filtering techniques. Filtering operations at the corporate's perimeter should generally be coarse-grained and not requiring complex parsing activity;
- *Organization-wide Policy Management.* There exists a clear separation of duties between corporate system administrators and application administrators with respect to the definition and management of access control policies. Attributes to be matched against a perimeter protection policy should be corporate-wide and specify application classes instead of being related to single services and single requestors.

3.3. Performance and manageability requirements

¿From the performance point of view, SOAP-based screening at corporate borders could become a severe communication bottle-neck. A list of requirements that should be taken into account when dealing with performance issues follows:

- simplicity of screening mechanisms (e.g. string pattern matching or simple regular expression matching), with the option of making them more complex and refined only at will;
- heavy usage of cryptographic mechanisms could easily become the dominating factor of processor workloads with consequent degradation of routing capabilities of network traffic;
- complex parsing techniques of the SOAP Body and network latency due to communication with third-parties like Security Token Services, Authorization engines or Certification Authorities are possibly sources of severe degradation of performance and routing capabilities;
- in real-world situations, communication efficiency is one of the most critical assets for business and, in practice, always takes the precedence over security. Security measures that may cause network degradation, are likely to be simply ignored.

As far as manageability is concerned, some functional requirements may be identified. Namely, firewall rules' evaluation should exploit:

- Grouping of requestors by common properties like affiliation and business role;
- Classifications of Web service interfaces in a descriptive structure based on the functions they offfer or on their location.

This way, a perimeter protection component could firstly parse some requestor's general properties, like the organization, the business role or network parameters, which are not application-specific. This lets a network or security administrator to fully define the perimetral policy with a little knowledge of application details and manage it in a (semi)independend fashion with respect to most modifications that application managers will make.

Then, the Web services requested, or the functional class of services to which the one requested belongs to, should be taken into account. Using a superset of the actual requested service, according to a classification to be specified, may help in terms of ease of both definition of the policy by the security administrator (i.e. he/she does not need to know the actual service names and meanings) and management (i.e. the administration task is (semi)independent from changes in service names or creations).

We shall develop on this descrition in the next section.

4. Design and implementation guidelines

4.1. Model

The solution proposed here is based on the results of the work by Damiani et al. [7] that presents a solution for an *Authorization Filter* for SOAP messages. Our architecture of the semantic-aware perimeter protection component has three main logical parts: a set of filtering rules representing the security policy and a mechanism to enforce it; a classification of requestors based either on groups or roles; and a functional classification of Web services.

The data structure that maintains the rules is stored persistently in XML and it offers an interface that permits to get a decision about allowing or dropping a SOAP message. It also provides other administration interfaces, for modifying the rules and for synchronizing the requestors and Web services classifications according to a common ontology that should be corporate-wide shared, between semantic-aware perimeter level filters and service level authorization filters.

Requestors Classification. Requestors, as said, should be grouped ac-

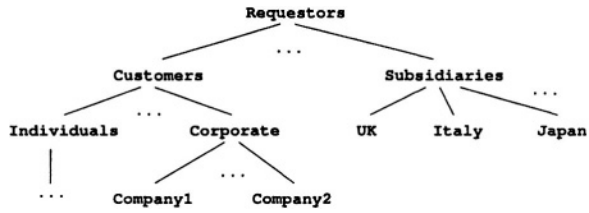

Figure 1. An example of requestor/organizational role classification

cording to some general and application-independent properties, based on business roles, affliations, etc. Figure 1 shows an example of requestor classification based on their organizational roles, such as *Customers* or *Subsidiaries, Individual* customers or *Corporate* customers and so on. Requestors classification represents the *subject* part of our authorization rules. Intuitively, more general groups (e.g. *Customers)* can be used for specifying broader rules (e.g. within an Internet banking context, "all customers could enjoy some services from the financial department"). More specific groups (e.g. *Company1*), instead, could be used for finer rules (e.g. "Company1 subscribed a trading on-line option, so Company1's requestors could enjoy the trading services of the financial department"). Individual requestors not falling in any group are treated as singleton groups.

Web services Classification. Web services, according to our proposal, should be classified into functional homogeneous categories and these categories, instead of single services, are used as authorization *objects.* This way, perimeter access control performs a filtering that is (semi)independent from specific services and based on corporate-wide criteria. In the example just mentioned describing the requestors classification (i.e. "Company1 subscribed a trading on-line option, so Company1's requestors could enjoy the trading services of the financial department"), we are ruling the access to a family of services called *Trading,* possibly grouping many actual services, like *PurchaseStock* and *ShowQuote.* Figure 2 gives an example of this Web services functional categorization.

4.2. Authorization syntax and semantics

Figure 3 presents the structure of authorization rules proposed by our model. Each rule is characterized by five attributes:

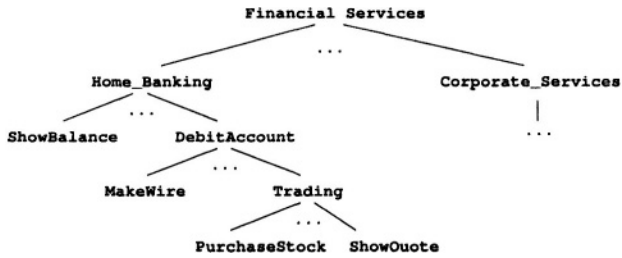

Figure 2. An example of service/functional category classification

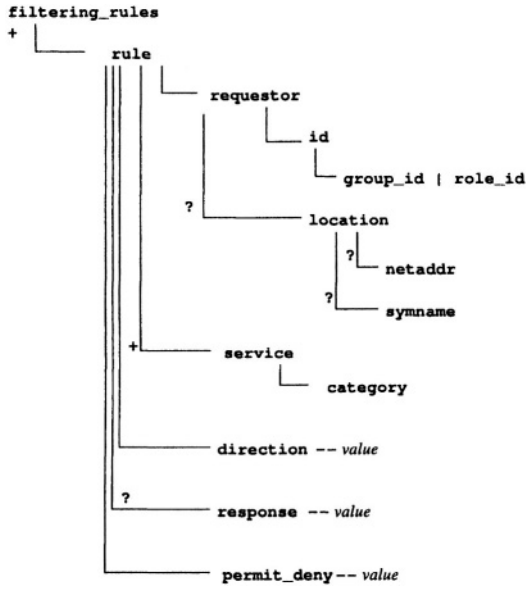

Figure 3. Filtering Rules structure

- *requestor,* the subject part of the rule, is a singleton (id) or a pair $(id, location)$, where id could be a group or a role defined within a requestor classification, and *location* could be a range of IP addresses or a symbolic name. For convenience, the value *ANY* should be specified when each requestor's id must be matched;

- *service,* the object part of the rule, takes values from the Web services classification. For convenience, the value *ANY* should be specified when each service must be matched;

- *direction* has the value *IN* for rules applied to incoming messages and the value *OUT* for rules applied to outgoing messages. Incoming SOAP messages could be requests from an external requestor towards an internal, protected Web service, or responses of an external Web service to an internal requestor, the opposite holds for outgoing messages. If the rule must be applied in both directions, for convenience the value *BOTH* can be set;

- *response* takes the value *YES* when the rule applies to a SOAP response only; it takes the value *NO* when the rule applies to a SOAP request only; and it is left undeclared when the rule does not depend on this information;

- *permit_deny* has the value *PERMIT* for positive authorization and value *DENY* for negative authorization. A positive authorization states that, for requestors described by *requestor,* the categories to which the requested services specified in the SOAP Body belongs to, described by *service,* are available and the message can pass through. A negative authorization, instead, states that *requestor* is not authorized to require any Web service belonging to *service,* then the SOAP message must be dropped.

4.3. Example of semantic-aware firewalling

Following the classification examples of Figure 1 and Figure 2, Table 1 presents an example of semantic-aware firewalling. The policy we want to enforce is: *All customers have a home banking account providing for some basic services like the account balance. Debit operations and on-line trading need specific banking accounts. Only Company1 has an on-line trading subscription for its personnel connecting from subnet 140.121.5.* For this example, we assume that the message exchange pattern is in the *request/response* form. Accordingly, the policy defined in Table 1 is composed by the following rules, to be evaluated sequentially from Rule 1 to Rule 6:

Rule 1 specifies that incoming SOAP requests (`direction value = "IN"`; `response value = "NO"`) from personnel of Company1 (`group-id Company1`), or sub-groups of, connecting from subnet 140.121.5 (`netaddr 140.121.5.*`), and requested services of the Trading category (`category Trading`), or super-sets of, are permitted to pass through (`permit_deny value = "PERMIT"`).

Rule 2 Specifies that Outgoing responses (`direction value = "OUT:`; `response value = "YES"`) to incoming requests ruled by Rule 1 are permitted.

Table 1. Example of semantic-aware perimeter protection policy for some home banking services

1	<requestor> <id> <group_id> Company1 < /group_id> < /id> <location> <netaddr> 140.121.5.* < /netaddr> < /location> < /requestor> <service> <category> Trading < /category> < /service> <direction value = "IN" /> <response value = "NO" /> <permit_deny value = "PERMIT" />
2	<requestor> <id> <group_id> Company1 < /group_id> < /id> <location> <netaddr> 140.121.5.* < /netaddr> < /location> < /requestor> <service> <category> Trading < /category> < /service> <direction value = "OUT" /> <response value = "YES" /> <permit_deny value = "PERMIT" />
3	<requestor> <id> <group_id> Customers < /group_id> < /id> < /requestor> <service> <category> DebitAccount < /category> < /service> <direction value = "ANY" /> <permit_deny value = "DENY" />
4	<requestor> <id> <group_id> Customers < /group_id> < /id> < /requestor> <service> <category> Home_Banking < /category> < /service> <direction value = "IN" /> <response value = "NO" /> <permit_deny value = "PERMIT" />
5	<requestor> <id> <group_id> Customers < /group_id> < /id> < /requestor> <service> <category> Home_Banking < /category> < /service> <direction value = "OUT" /> <response value = "YES" /> <permit_deny value = "PERMIT" />
6	<requestor> ANY < /requestor> <service> ANY < /service> <direction value = "BOTH" /> <permit_deny value = "DENY" />

Rule 3 says that generic Customers, or super-groups of, cannot access services in the DebitAccount category or sub-sets of. Both requests or responses, either incoming or outgoing (`direction value = "ANY"`), are denied and hence dropped (`permit_deny value = "DENY"`).

Rule 4 says that incoming requests from generic Customers or sub-groups of, requesting services of the Home_Banking category, or super-sets of, are permitted to pass through;

Rule 5 specifies that outgoing responses to incoming requests ruled by Rule 4 are permitted.

Rule 6 is a *default rule* stating that if any of the previous does not applies, then we want to drop the messages.

5. Conclusion and Future Work

In this paper we gave a functional description of a semantics-aware firewall, a perimeter protection component matching a requestor's general properties and the functional class of requested Web services against a locally stored access control list.

The benefits of our approach are several

- the solution is scalable by designing filtering rules of increasing complexity (i.e. simpler and faster controls for direct Internet traffic for a first layer of filtering, possibly followed by more detailed controls for messages already screened);
- the solution can be customized according to the needs of specific businesses;
- untrusted requestors do not interact in an unfiltered fashion with Web services, which are highly critical components;

Moreover, the perimeter protection solution described can be smoothly integrated with the solution for fine-grained access control for SOAP Web services developed in [7] by some of the authors, composing this way an overall comprehensive approach to SOAP Web services security issues.

However, the solution proposed is not complete and many issues are still open. Other more complex message exchange patterns should be considered. Considering filtering techniques, we plan to extend our proposal to *stateful* semantic-aware firewalls by correlating messages exchanged according to HTTP and SOAP protocol semantics. Another important issue for our work is the full integration between this semantic-aware perimeter level and the fine-grained access control level, and the development of solutions for coordinating the two levels of security policies.

References

[1] Atkinson, B. and et al. Web services security (ws-security), April 2002. http://msdn.microsoft.com/ws/2002/04/Security.

[2] Box, D. and et al. *Simple Object Access Protocol (SOAP) 1.1*. World Wide Web Consortium (W3C), May 2000. http://www.w3.org/TR/SOAP.

[3] Bray, T.,Paoli, J., Sperberg-McQueen, C.M., and Maler, E. *Extensible Markup Language (XML) 1.0 (Second Edition).* World Wide Web Consortium (W3C), October 2000. http://www.w3.org/TR/REC-xml.

[4] Brenton, C. and Hunt, C. *Active Defense: A Comprehensive guide to network security.* Sybex, 2001.

[5] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. and Samarati, P. Controlling access to XML documents. *IEEE Internet Computing,* 5(6):18–28, November/December 2001.

[6] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. and Samarati, P. A fine-grained access control system for XML documents. *ACM Transactions on Information and System Security,* 5(2):169–202, May 2002.

[7] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. and Samarati, P. Securing SOAP e-services. *International Journal of Information Security (IJIS),* 1(2):100–115, February 2002.

[8] Damiani, E., De Capitani di Vimercati and Samarati, P. Towards Securing XML Web Services. In *Proc. of the 2002 ACM Workshop on XML Security,* Washington, DC, USA, November 2002.

[9] Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T. Hypertext Transfer Protocol – HTTP/1.1, June 1999. http://www.ietf.org/rfc/rfc2616.txt.

[10] Graham, S. and et. al. *Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI.* Sams Publishing, 2002.

[11] Kudo, M.and Hada, S. XML Document Security and e-Business applications. In *Proc. of the 7th ACM Conference on Computer and Communication Security,* Athens, Greece, November 2000.

[12] Mitra, N. *SOAP Version 1.2 Part 0: Primer.* World Wide Web Consortium (W3C), May 2002. http://www.w3.org/TR/soap12-part0/.