A REQUIREMENT FOR A XML WEB SERVICES SECURITY ARCHITECTURE

Luminita Vasiu, Cristian Donciulescu
Middlesex University, London, UK

Keywords: XML Web Services, Internet security, Web Services Architecture

Abstract: Lately, XML Web Services are emerging as a dominant platform in the computing world. At the moment, the web has evolved into an active medium for providers and consumers of services**.** One of the major problems for XML Web Services is the related with security. The paper describes a comprehensive XML Web Services Management Architecture that supports, integrates and unifies several security models, mechanisms and technologies in a way that enable a variety of systems to securely interoperate in a platform independent, comprehensive manner.

1 SECURITY ISSUES SPECIFIC TO XML WEB SERVICES

In order to provide a comprehensive security model for Web services, it is required to integrate the currently available processes and technologies with the evolving security requirements of future applications. There are a big number of similarities between security issues raised by a normal web environment and those posed by a XML Web Services Environment. However, due to the particular nature of XML Web Services, some of these issues are more complex and new problems arise. As the technology becomes adopted on the market on a wider scale, there will be a need to integrate more and more services under the same environment. This integration, which has to be seamless in order to take advantage of what XML Web Services have to offer, poses the biggest security problem. Different security contexts must be able to talk to each-other without any security compromise.

The XML Web Services environment has the following defining characteristics (Yang, 2002):

- It is highly decentralised in architecture and administration;
- It is implemented in different technologies, from different vendors and with different tools;
- It is spread over multiple enterprises, contexts, etc;
- It must have parts that are accessible to everybody on the Internet and parts that are restricted to authenticated users. These parts often collaborate with one-another;

In the recent period, there have been many malicious attacks on several high-profile Web servers, hosting applications used by millions of computers throughout the world. Attacks like those performed by Nimda, Code-Red, MSBlaster and Sobig have produced huge damages and there are fears that due to their specific nature, XML Web Services will become favourite targets for such attacks if they will not be properly protected.

As with regular web applications, major areas that have to be addressed to achieve a secure environment are authentication, access control, data privacy (encryption), non-repudiation, etc. In each of these areas there are specific problems to be addressed when dealing with XML Web Services (Stencil Group, 2003). The main issues are discussed briefly below:

- Authentication. In regular web applications, usually clients have to be authenticated when requesting to use a service. This is usually done using passwords, certificates or infrastructures like Kerberos and LDAP. However, in the case of a XML Web Service, the server has to be identified to the client as well, considering that in a SOAP request/response pair, both messages (including the request) can contain very sensitive information. That would require

Vasiu L. and Donciulescu C. (2004).
A REQUIREMENT FOR A XML WEB SERVICES SECURITY ARCHITECTURE.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 60-66
DOI: 10.5220/0002619400600066

the XML Web Service to have a security certificate attached to its contract that can be interrogated by a client before sending a request, to prevent talking to a spoofed XML Web Service.

- Access control. With XML Web Services, controlling authorisation rights is a difficult task, because not only the client has to have access to the specific service, but also to the specific operations that service performs. There can be instances where different levels of access to a service's functions must be assigned to users. It is also difficult to monitor attempts of illegal use of the service, since usually these types of applications serve other applications, so there is no human involvement in the entire process. The administration of such an environment with multiple types and levels of access rights is not an easy task. There are issues here as well (Yang, 2002), considering that having a single administrator over the entire web service environment is a security threat in itself. Having multiple administrators with different privileges is an option, but this triggers the need for an administrative platform for XML Web Services. And due to the very diverse nature of XML Web Services, such a platform should be very open, allowing for all kinds of security policies to be integrated in it, as well as multiple and diverse administrative schemes.
- Data privacy. A SOAP message between a client and a service can be encrypted to protect its content, however, unlike regular web applications, where usually a client and a server communicate with each-other and the client trusts the server, with XML Web services, the trust has to be reciprocal. Also, although a client may trust a service and send sensitive encrypted content to it, the service may act as a client to other services. How can a client trust that a service will protect their information properly while communicating to other services?
- Digital signatures. To ensure data integrity between clients and web services digital signatures have to be employed. Fortunately, the XML Signature standard has been designed exactly for that purpose. Digital signatures also provide a non-repudiation mechanism, to verify that an action really took place.

2 CURRENT SECURITY STANDARDS

The expectations of the developer community with regard to XML Web Services technology and its implications on the development of the Internet were very big when the WS specification was first made public two years ago. There were voices that argued XML Web Services to be the "biggest step forward in the development of the Internet since the advent of the browser"(Iona Software, 2001).

However, although there is still great enthusiasm in the developer community and there are attempts made by big companies to use the technology, the development expected in the beginning is far behind what was imagined. That is mainly because the original XML Web Services standard did not include any security specification; so serious projects using XML Web Services were not undertaken. Web Services have been used until now mainly in small undertakings and in applications designed to run behind company firewalls and intranets. The few examples where big companies used Web Services for their commercial (such as British Telecom in the UK) products rely on custom security implementations. Out of experience, custom security implementations are much more subject to errors and security holes than standardised security platforms.

To cover the lack of security specifications in the original Web Services platform, several important companies are working together, trying to develop a security specification for XML Web Services. The results are several standards that are either in the final stages of development or have been recently approved by organisations such as OASIS or W3C.

However, as key personalities involved in the development of these standards readily admit, "the implementation of a particular spec will [not] make everything completely interoperable and secure" (Maler, 2003). In order to have a (reasonably) complete secure XML Web Services environment, several standards must be implemented. This shows the need for the development of security architectures that combine the standards into a comprehensive security framework.

- **WS-Security:** The main security specification that deals with the security of XML Web Services is WS-Security, developed by a collaboration between Microsoft Corporation, IBM and Verisign Inc., under the coordination of OASIS. WS-Security is the most comprehensive security standard to-date defining how to manage and coordinate aspects such as message integrity, authentication and

confidentiality and specifying how to include features such as digital signatures and encryption in SOAP documents (Stanton, 2003). Although dealing with many important aspects in defining how security architecture should work, the specification fails to address other vital issues with XML Web Services, such as fail-over, backup, redundancy, etc. However, this standard is the basis for a security infrastructure that is being developed by the same organisation. This security infrastructure contains other developing standards, such as: WS-Policy, WS-Trust, WS-Privacy, WS-SecureConversation, WS-Federation, WS-Authorisation. These standards are now in a drafting stage and are not yet being used in security implementations.

- **Shortcomings of current standards**. With all the standards being proposed, drafted and approved, there are many people that feel that developers will soon face another hurdle: making all of them work together smoothly. And still, there are unanswered questions regarding many issues with security. How will and outside, potentially weak, system be integrated within an already secured environment? How do you perform a security audit, to test that a XML Web Services environment is secure, when it is spread over different networks and administrative domains? (Yang, 2002) What happens to its clients when a Web Service is offline, either for maintenance or due to a malicious action? How will the applications using that particular service work around the problem? How does someone prevent malicious clients from initiating attacks on a XML Web Service? There are fears that XML Web Services, in particular high-profile ones, will become favourite targets for attackers. There is no way at the moment to discover a XML Web Service dynamically and there is also no way of identifying web services that perform identical tasks. It becomes obvious that an attack on a XML Web Service, which might very well serve thousands or millions of clients, would become much more damaging that present-day attacks on web application servers, which only have the potential to take down applications running on that particular server. To address these particular issues, related specifically to XML Web Services, a specific kind of security architecture would have to be designed.

3 ATTACKS ON WEB SERVICES

The main benefit of the XML Web Services technology is that it exposes functionality, sometimes critical functionality through standard HTTP protocols. XML Web Services traffic usually flows through port 80, such as regular HTTP traffic. However, as opposed to HTML and regular web sites, the functionality exposed by a XML Web Service is much more sensitive. Firewalls can be configured to intercept SOAP traffic, but most of them can only block it. If you allow SOAP traffic through the firewall, there is no way to interpret the contents of the SOAP messages outside of the XML Web Service itself, thus a malicious request can easily go undetected.

Also, and even more dangerous, XML Web Services are self-descriptive. The contract (WSDL) file of the service contains detailed information about the behaviour of the service, from which an attacker can know exactly what the vulnerabilities of the service are. This makes attacks much more likely to succeed than on regular web applications.

Unencrypted or poorly encrypted, SOAP messages can easily be intercepted and tampered with. For these reasons, until now, there are no big-scale implementations using XML Web Services. The following types of attacks are most likely to be performed on XML Web Services:

- Denial of service. This is a common attack on a regular web server that will take the server down immediately. An attack on a XML Web Service will take down the Web Service and the applications that use it. Such an attack on a XML Web Service is obviously much more damaging than on a web server. The problem is even bigger than it seems (Yang, 2002), because in the case of Web Services a firewall cannot always handle the problem. Due to the fact that a XML Web Service might work a lot to satisfy a single request, the service can be easily overwhelmed by legitimate and not fake requests. This will still take down the service and the firewall cannot prevent it without denying legitimate users from accessing the service.
- Replay attack. A replay attack consists in recording legitimate requests to the XML Web Service and then transmitting them over and over again to the service. However, this kind of attack can be detected easier by analyzing and denying identical requests to the XML Web Service inside a safety time interval. Still, this attack, combined with a

distributed denial of service attack will be very difficult to contain.

- Buffer overflow. A buffer overflow attack consists of sending a request longer in size than the service is expecting. This can cause the service to crash, or, worse, allows the attacker to execute code on the server machine, with disastrous results. Due to the fact that XML Web Services are ideal to expose functionality offered by old legacy systems that do not have any protection to this kind of attack, there might be a big problem with protecting such systems using XML Web Services from it.

4 THE XML WEB SERVICES MANAGEMENT ARCHITECTURE (WSMA)

XML Web Services are very complex and they have unlimited application areas. In this context, it is very difficult to create a comprehensive management architecture that should be easy to administer and maintain throughout multiple domains, security policies and administrative rights (Stencil Group, 2002).

A Web Services Management Architecture (WSMA) should be able to handle administrative tasks without any need for customizing it for specific clients or web services:

Some of the requirements imposed for the WSMA are:

- To control authorisation of incoming requests;
- To maintain access rights for clients;
- To verify the identity of a client;
- To allow the deployment of a XML Web Service without any downtime;
- To manage situations where a Web service may become unavailable;
- To manage data integrity tasks such as encrypting and signing outgoing messages;
- To maintain information about the functionality of managed Web services;
- To be able to inherit permissions and settings from other trusted environments and have the possibility to trust other environments.

The WSMA architecture will create a sort of a "bubble" around the XML Web Services managed. It will be able to isolate them from the outside world and act as a kind of intermediary between the clients and the managed services. This could be accomplished by a mechanism similar to a firewall that intercepts all incoming SOAP requests. When receiving a message, the WSMA will perform the following steps:

- Decrypt the message using a private key (assuming the client had the message encrypted);
- Verify the identity of the client;
- Identify the service for which the client made the request. If the service resides within the current context, verify if the client has the rights to access that service (or even the particular method within the service).If the service is not found within the current context the architecture should pass the SOAP request to all the other trusted contexts;
- Allow the service to run or wait for the other contexts to return a response and grab the result (the response SOAP message);
- Encrypt the response for the client if necessary;
- Send the response back;

The above sequence of events is based on the following assumptions:

- There is a possibility to administer the individual local security context completely independent from the others;
- The local context can inherit all or some security settings from another context and from only one other context;
- The local context can trust any number of other contexts and can lend some or all its settings to those contexts;
- There is a mechanism that allows a client to address a request for a service to a parent context that will redirect it to its destination. Such a mechanism is possible to be established by providing each context with the possibility to publish the WSDL contracts of the XML Web Services managed on its parent.

The architecture described above is in fact a tree of contexts that trust their parents and can be managed locally by independent administrative entities. Each context can manage any number of XML Web Services. A context has total control over all the services it manages. The administrative entity of a context it is able to manage at least the following aspects for the services it controls:

- Two configurable levels of authentication for clients: at the XML Web Service's level and at the method level for each particular XML Web Service;
- Whether or not the SOAP requests should be accepted unencrypted for each particular managed XML Web Service;

- Whether the SOAP responses should be sent encrypted all the time, only on request, only to specific clients or never
- the possibility to build contingency plans for the situation when a XML Web Service becomes unavailable.

These plans could include a standard SOAP response that the client can handle within the parameters expected for each request or the possibility to specify alternative hosts for the same XML Web Service or even alternative services performing the same task. This could be taken forward in the future by creating a possibility to specify equivalence between different XML Web Services that perform similar tasks. For now, specifying alternative services could be accomplished by building interfaces between them. An interface language based on XML of course could be beneficial at this point. The context should allow adding, removing or modifying information about the managed XML Web Services on-the-fly, without the need to take the context down at any time.

be a skeleton of the original service and it would be under the total control of the parent context.

Therefore, apart from the XML Web services each context manages, it would also manage a number of special Web Services that represent the trusted ("child") contexts' XML Web services. Because each of these particular Web Services are part of the "parent" context, the administrative entity has close control upon the activity of each "child", thus on the level of trust each "child" is allowed to have. This in turn allows for a unified management scheme both for the XML Web Services managed and for the other web management contexts. Such architecture is presented in Figure 1.

The architecture presents three management contexts with their managed XML Web Services represented by ovals. WSMA 2 and 3 trust WSMA 1. Each WSMA has an access control list (ACL) with the permissions specified in parenthesis. Thus, ACL 1 gives full access to Client 1 and ACL 2 allows full access to Client 2.

There are three scenarios (A, B, C) depicted in the diagram. Each scenario is represented by a set of

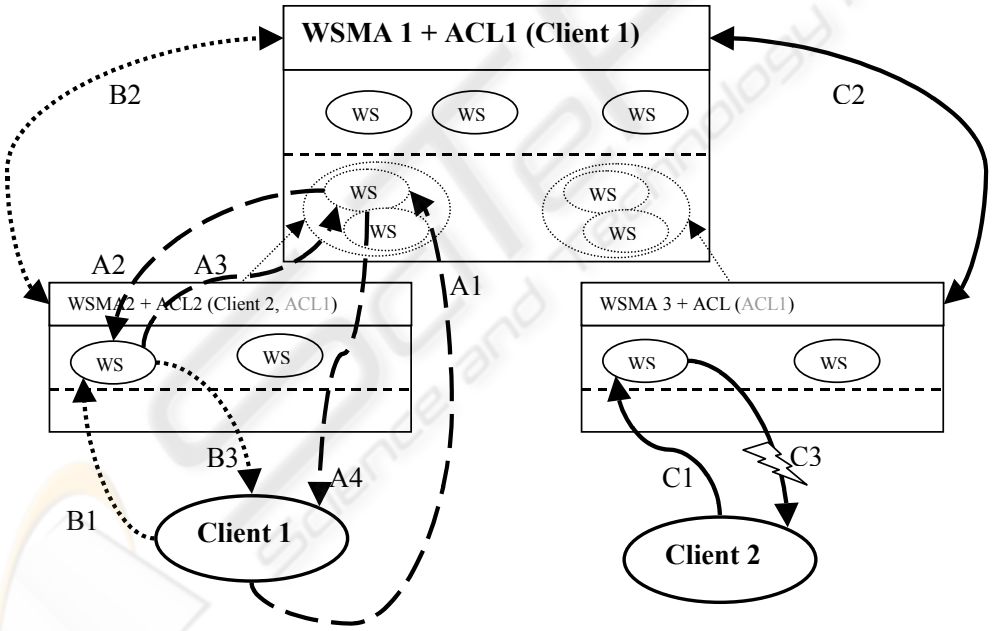

Figure 1 - Web Services Management Architecture (WSMA)

A possible solution for allowing contexts to trust one-another thus allowing for a trust tree to be built is based on providing each management context with the possibility to build a proxy XML Web Service for itself or for each of its managed XML Web Services. These proxy XML Web Services would have to be automatically published by the WSMA on the parent context, without any outside intervention. At the moment, the second alternative seems more feasible. Each proxy service would just

different styled arrows and the associated actions are numbered with the scenario letter followed by a number representing the order in which the action takes place.

- **Scenario A**: Client 1 makes a request to a Web Service (A1) located in WSMA 2. However, the client addresses the request through WSMA 1 to which it has full access rights given by ACL 1. The client is therefore authenticated via ACL1

and its request is allowed to go to the destination Web Service. The service it requested is only a proxy in WSMA 1, so the proxy will call the corresponding Web Service in WSMA 2 (A2). Since the request comes from WSMA 1, that WSMA 2 trusts, the request is let through and the business process executes. The response is relayed back to the proxy in WSMA 1 (A3). The proxy responds to the client with the SOAP message expected (A4).

- **Scenario B**: In this case, Client 1 has called directly the service running in WSMA 2 (B1). However, Client 1 has no local rights specified in ACL 2, therefore the request cannot be allowed through. WSMA 2 will interrogate its parent context (B2) to check the access rights from ACL 1. ACL 1 gives full access to Client 1, therefore Client 1 has full access rights on WSMA 2 (since there is no other rule that overrides the inherited ACL1 rule). Client's 1 request is then cleared to run in WSMA 2. The Web Service called will respond to Client 1 (B3).
- **Scenario C**: Client 2 makes a request to a Web Service managed by WSMA 3 (C1). Client 2 has no access rights specified in ACL 3 so WSMA 3 will interrogate its parent (WSMA 1). ACL 1 gives no rights to Client 1 and since WSMA 1 has no other parent to which to relay the authentication request, it will answer to WSMA 3 that Client 2 has not been authenticated (C2). At this stage, WSMA 3 will refuse to execute the request placed by Client 2.

Some of the advantages for the WSMA are presented bellow:

- It allows for complete delimitation of administrative rights over local XML Web Services
- It allows the possibility for a client to have access to an entire environment or business process rather than having access to each service that is part of that business process;
- It allows a chain of trust to be built, which in turn allows a client to have access to XML Web Services for which specific security settings have not been specified;
- It is completely separated from the XML Web Services implementation, thus being able to handle any service;
- There is no single administrative entity that manages security issues;
- A service can be easily plugged into a management context without affecting its functionality;
- It allows for alternating Web services to be specified by a management context, thus creating a way to handle situations where the main service is unable to respond;
- A load balancing mechanism can be easily implemented because the SOAP messages are intercepted by the management architecture and dispatched to the destination XML Web Service;

However, there are a few open questions about the WSMA. It is unclear how WSMA would behave when the trust tree becomes big. The main problem is that once the trust tree gets beyond a certain size, publishing a new XML Web Service in the WSMA would require a big effort of propagating the new Web Service contract up the tree. However, it is not clear now what the procedure for publishing a new Web Service would be, so it is impossible to establish what the feasible June 2001 size for a WSMA is (Stencil Group, 2003).

There is still uncertain whether the XML Web Services visibility system within management contexts should be simple (visible externally/not visible externally) or a more complex one, such as the one used for setting member visibility in object-oriented programming (public/private/protected – with the same meaning as in OOP).

Further testing will take place to establish various mechanisms and standards required to implement the architecture requirements.

5 CONCLUSION

A key benefit of the emerging XML Web Services architecture is the ability to deliver integrated, interoperable solutions. Ensuring the integrity, confidentiality and security of Web services is crucial both for the organisations and their customers. By leveraging the natural extensibility that is the core of the XML Web services model, the specification build upon functional technologies such as SOAP, WSDL, XML Digital Signatures, XML Digital Signatures, XML Encryption and SSL/TLA allows XML Web service providers and users to develop solutions that meet the individual security requirements of their applications. The paper has presented a XML Web Services management architecture that is able to handle administrative tasks without any need for customizing it for specific clients or Web Services. The architecture presented has definitely a lot of advantages such as allowing the complete delimitation of administrative rights over local Web services, the possibility of building a chain of trust,

the capability to handle any services and so on. However, further work will be carried on in order to prove the complete viability of the WSMA.

REFERENCES

Iona Technologies website, 2001, http://www.iona.com

Maler E. inverview, 2003 - http://java.sun.com/features/2003/03/webservices-qa.html

Stanton, D., 2003. "The Differentiation Of Web Services Security" – Web Services Journal

Yang, A., 2002. "XML Web Services Security Issues" – article, http://www.xwss.org/articlesThread.jsp?forum=34&thread=648

The Stencil Group, 2003. "Web Services Rules: Lessons learned from early adopters"

The Stencil Group, 2002. "Understanding Web Services management"

Robins B., 2003. "There is no Web Services yellow brick road" http://searchwebservices.techtarget.com/originalContent/1,289142,sid26_gci883333,00.html

Thelin, J., 2002. "Web Services and remote references – an intimidating task?", http://www.webservicesarchitect.com/content/articles/thelin01.asp

Samtani, G, Sadhwani, D, 2003. Web Services and Peer to Peer computing", http://www.webservicesarchitect.com/content/articles/samtani05.asp

Ozu N., Duckett J., Watt A., Mohr S., Williams K., Gudmundson O. G., Marcus D., Kobak P., Lenz E., Birbeck M., Zaev Z, Livingstone S., Pinnock J., Visco K., May 2001. "Professional XML (Programmer to Programmer)", Wrox Press, ISBN:1861005059

Jorgansen D, June 2001: "Developing Web Services with XML", Syngress, ISBN: 1-928994-81-4