An Architecture for An XML Enabled Firewall

Andrew Blyth
School of Computing, University of Glamorgan, Pontypridd, CF37 1DL, UK
(Email: ajcblyth@glam.ac.uk)

(Received Mar. 14, 2006; revised May 25, 2006; and accepted July 31, 2006)

Abstract

XML is rapidly becoming the default way for organizations to sharing information across networks and organizational boundaries. XML was designed as an information mark-up language and was not designed with security in mind. Consequently we are left with the problem of security XML documents from attacks such as malicious modification or fabrication. With modern VPN technology such as SSL we can encrypt and secure a data stream as it cross the network. However, when the data stream encounters an organizational boundary such as a firewall the XML document has to be parsed and forwarded to any back-end systems that require it. Current IDS and Firewall Technology does not have the ability to identify when the contents of an XML document is being attacked. This paper outlines a firewall architecture for the secure exchange of information using the extensible mark up language (XML). The architecture can be used to create a virtual private network suitable for an e-commerce application, allowing secure communication over the Internet. This paper identifies the elements required to build an XML enabled firewall that will (1) ensure the secure communication of data, and (2) validate the data to ensure data integrity. The architecture addresses the issue of information integrity using the Document Type Definition and additional rules applied by a proxy.

Keywords: Firewalls, information security, network security, XML

1 Introduction

Increasingly organizations are connecting their Intranets to the Internet through firewalls and creating virtual private networks. This requires various types of information system to communicate in a secure manner. Many organizations use firewalls to provide a level of security by controlling access to information systems. By linking firewalls together via encrypted network connections it is possible to create a virtual private network [11]. A virtual private network is a network where packets that are internal to a private network pass across a public network, without this being apparent to users and hosts on the private network [2]. Most organizations either have or are planning to make use of B2B or C2B technology, and modern B2B or C2B technology makes use of XML.

The objective of the XEON project was to create a firewall architecture that would allow various information systems located behind a firewall, including legacy systems, to communicate and interoperate, via the exchange of XML documents with, each other in a secure manner across unsecured networks. The solution makes use of the extensible mark up language - XML [8], as a wrapper that will allow the encapsulation of data. The XML document can then be tunnelled between firewalls via proxy services that utilise the HTTP, SSL and HTTPS protocols. Thus from the perspective of an information system located behind the firewall, remote systems on the Internet will appear to be systems on the local Intranet. From an intruder's perspective all that can be seen are two firewalls communicating with each other via HTTP over an encrypted channel.

In the following sections this paper will discuss how XML can be used as a data wrapper and how proxy services can be used to create the impression of multiple servers located behind a firewall. It will then define in detail the XEON architecture that allows this impression to be created. In defining the architecture how the firewall implements security policies in order to create the virtual private network and how data integrity can be validated will be examined.

2 Information Integrity through XML

XML is emerging as the key standard for information interchange in e-commerce applications over the Internet [4]. XML allows users to define their own mark up vocabulary for describing, in a rich and machine understandable way, the documents they wish to manipulate. The rules that define the mark up vocabulary take the form of a Document Type Definition (DTD). This DTD provides a way of validating a marked up document of a specific type against the 'approved' definition for that document type. The DTD may be external to the XML document, embedded within the document, or both. The most fundamental type of checking that can be performed on an XML document is to determine whether or not it is well formed. A well-formed XML document conforms to the W3C's XML 1.0 Recommendation [8]. Whilst this checks that the document is an XML document it makes no reference to the DTD. The process of checking an XML document against a DTD is known as validation. In addition to checking that the document is well formed it is also checked for conformance to the rules encoded in the DTD. Within a DTD it is possible to define elements, element nesting, optionally of elements sequences of elements, cardinality of elements and element attributes. A variety of useful functionality is however lacking, including the notion of data types (therefore no type checking can be applied). A DTD also provides no mechanism for expressing class/subclass relationships, inheritance or cross element constraints.

It was demonstrate in [1] that XML security standards such as [6] that relay on the integrity of the XML document can be subverted via the embedding of a DTD inside the XML document this over riding of the DTD security signature document. Even the used of security roles and rights such as [7] can be subverted. The reason that these security standards are vulnerable is that they all relay on the internal integrity of the XML document and the ability of an XML document to make an external reference to a DTD. It was shown on in [1] that such security features can be over-written and subverted.

Thus, while XML can provide some degree of security and validation checking, it is not in itself sufficient. XML Schema extends the functionality provided by the validation, but still has significant weaknesses for instance in cross element checking. The checking and validating between elements in an XML document is particularly important when used as part of an e-commerce solution.

3 Proxy Servers

A proxy server for a service is an application that from the perspective of the Internet behaves like the specified service, The proxy server activity passes complete messages to the application server located behind the firewall on the corporate Intranet. When combined with encrypted channels, proxying can be used to create a virtual private network [2].

- From the perspective of a server located behind the firewall, the firewall behaves like the target application server with which the server is trying to interact.
- From the perspective of a client located in front of the firewall, the firewall behaves like the target application server with which the server is trying to interact.

The advantage of a proxy server is that it has the potential for performing pre-processing on the complete request and decides whether or not to forward the request to the application server. If the request is approved then the proxy server will interact to the application server on behalf of the information system program. As far as the user is concerned, interacting with the proxy server is just like interacting with the application server. Because proxy servers understand the underlying protocol and application, they allow logging of relevant security information to be performed in a particularly effective manner.

One disadvantage of using a proxy service is that the proxy services lag behind non-proxied services from a performance perspective. The proxy services may also require different servers for each service, can require modifications to clients and are not workable for some services [2].

There are a number of proxy servers available (both commercial and free-ware), however none of these servers provide the ability for security an XML document before passing it onto a back-end e-commerce system. The goal is XEON is to security such document before exposing such back-end systems.

4 The Xeon Architecture

The purpose of the XEON architecture is to provide a mechanism through which various applications (such as e-commerce systems) can have their application level protocols encoded via XML and transferred securely to another application. The ability of the architecture to function in a transparent manner from the perspective of an application allows for the creation of a virtual private network. A high level overview of the proposed architecture can be found in Figure 1.

The three top-level elements of the architecture are User, Firewall and Application. The User is the external agent interacting with the Application. The User may be a person or a system. The User can be identified by their IP address, denoted by UIP in Figure 1. The Firewall is responsible for managing the interaction between the User and the Applications behind the Firewall. The Firewall can be identified by its IP address, denoted by FIP. The Application is the system that the User is interacting with. The Application can be identified by its IP address, denoted by SIP. The SIP is masked from the User who is aware only of the FIP. Although Figure 1 shows only one Application there would typically be a number of Applications behind the Firewall.

The Firewall itself consists of four elements:

- **Network Manager**. The Network Manager is responsible for routing packets between the User who is located in front of the Network Manager, and the Firewall.
- **HTTP Proxy**. The HTTP Proxy will route XML traffic from the Network Manager to the XML Router and non-XML traffic to the appropriate Application. (The precise nature of the Application in handling non-XML traffic is beyond the scope of this paper.) It will route XML traffic from the XML router to the Network Manager.
- **XML Router**. The XML Router routes incoming XML traffic to the appropriate Application Proxy.

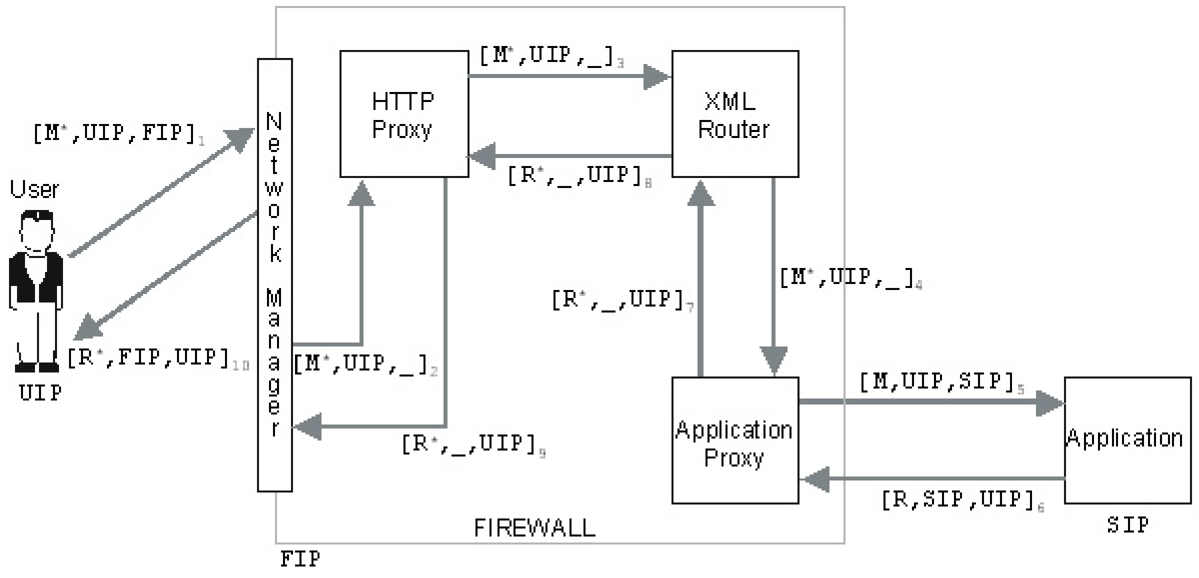

Figure 1: The XEON architecture

The identification of the appropriate Application Proxy is based on the public DTD of the XML document. It routes outgoing XML traffic through the HTTP Proxy for security reasons. The XML router strips out public DTDs from incoming documents and places the equivalent secure DTD within the XML document. For a given DTD the public and secure versions should be identical. For outgoing documents the XML proxy strips out the secure DTD and replaces it with the equivalent public DTD. This dual DTD approach is included as a security measure.

- **Application Proxy**. The Application Proxy is specific to a DTD, but there may be a number of application proxies for a given Application. The Application Proxy is responsible for parsing and applying the specified validation to an incoming XML document and communicating the contents of that document to the Application in an appropriate format. Thus legacy applications can have their message wrapped inside an XML document using a DTD. This document can then be transported in a secure fashion across the Internet and unwrapped at the other end. Outgoing messages from the Application are wrapped as an XML document and passed to the XML Router.

Each of these elements is described in greater detail below. The passage of messages through the architecture is indicated in Figure 1 by the arrows and the composition of the messages by the triples in square braces. The conventions used in the triples are:

$$\texttt{[Document, From-IP, To-IP]}_{\text{index}}$$

Document - indicating the direction and format of the document:

M*: An incoming document in XML format.

M: An incoming document in appropriate native Application format.

R: An outgoing document in native Application format.

R*: An outgoing document wrapped in XML.

From-IP - indicating the sender of the message:

UIP: The User IP address.

SIP: The Application IP address which indicate the From-IP has been stripped out.

FIP: The Firewall IP address.

To-IP - indicating the destination of the message:

`FIP`: The Firewall IP address which indicate the To-IP has been stripped out.

`SIP`: The Application IP address.

`UIP`: The User IP address.

index - The index is simply used in the figure to indicate the sequence of steps that a User message and the Application response to that message follows.

Referring to Figure 1, it is possible to trace the path of a successful User interaction with an Application:

1) The User message containing an XML format document `M*`, the Users IP address `UIP` (the From-IP) and the Firewall IP address `FIP` (the To-IP) is received by the Network Manager.

2) The Network Manager strips out the `FIP` from the To-IP and passes the message to the HTTP Proxy. The Network Manager also applies a set of firewall specific rules and checks to the incoming connection.

3) The HTTP proxy identifies the document as being in XML format and passes the message to the XML Router.

4) The XML Router identifies the appropriate Application Proxy by examining the DTD of the XML document and passes the message to that Proxy. In addition, the XML Router will:

   a. Strip the public DTD out of the XML document and insert the equivalent secure DTD into the document.

   b. Apply the incoming security policy to the XML document (see Figure 2).

5) The Application Proxy parses the XML document, performing validation if specified. It performs any application specific checks and converts the result of parsing the document into a format appropriate to the Application. This converted document in native Application format is then passed to the Application.

6) The Application returns its response `R` to the Application Proxy. This is achieved by the application interacting with the application proxy using its own high-level application specific protocol, for example, SMTP.

7) The Application Proxy wraps the Application response in XML and strips out the Application IP address `SIP`, passing the resulting message to the XML Router.

8) The XML Router will: a) Apply the outgoing security policy associated with the secure DTD to the XML document (see Figure 2), b) Strip the secure DTD out of the XML document and insert the public DTD into the document, and c) Once the security policy has been applied and the secure DTD has been replaced by the equivalent public DTD, the XML Router passes the message to the HTTP Proxy.

9) The HTTP Proxy passes the message to the Network Manager.

10) The Network Manager places the Firewall IP `FIP` into the From-IP field of the message and passes the message to the User IP address `UIP`.

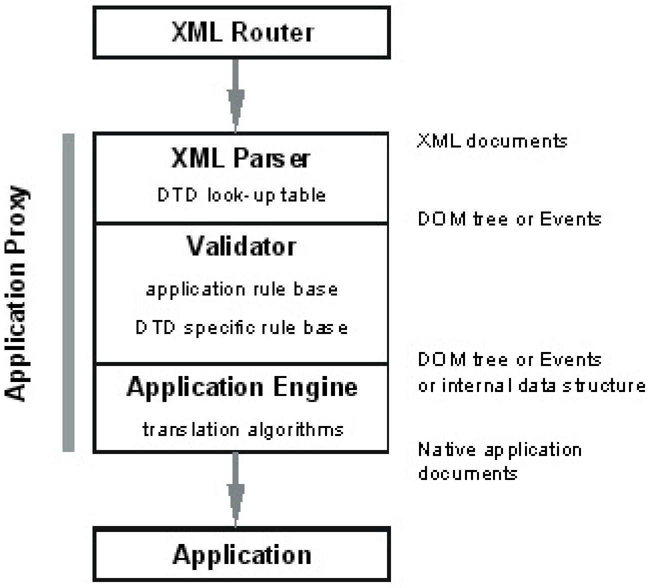

Figure 2: The XML application proxy (incoming)

4.1 Network Manager

The purpose of a firewall is to restrict access between a protected network and the Internet, or between other sets of networks [11]. The network manager will perform packet filtering based upon a set of rules applied to any network connection. These rules define the security policy of the firewall. Typically a network manager will make a decision to accept packets based upon the IP source address and IP destination address, the protocol (whether the packet is a TCP, UDP or ICMP packet), the TCP or UDP source and TCP or UDP destination ports, and the ICMP message type. In addition to the information included in the packet header, the network manager will also have access to the network interface the packet arrives on and the network interface the packet will go out on. Examples of the ways in which a firewall might be programmed to selectively route packets to or from a site include:

- Block all incoming connections from systems outside the internal network except incoming SMTP connections.
- Block all connections to or from certain untrustworthy systems.
- Allow e-Mail, FTP services, but block dangerous services such as TFTP, RPC, etc.

There is a known set of vulnerabilities inherent within the TCP/IP protocol definition, such as SYN flooding and IP spoofing [5]. It is the responsibility of the firewall to manage and respond to all of the vulnerabilities contained in the TCP/IP protocol suite.

4.2 HTTP Proxy

A major concern regarding the security of HTTP proxies is the degree of damage that can be caused to the HTTP server by a malicious client. The security counter measures that are required to resolve the security concerns are well understood and well documented [11]. The HTTP proxy within the XEON firewall simply functions as a normal HTTP proxy with one exception, this is that when an XML document arrives it is forwards to the XML router. In addition, the HTTP proxy will also manage the secure point-to-point communication via the use of the secure socket layer (SSL). Typically HTTP functions through port 80 and the secure socket layer functions through port 443.

4.3 XML Router

The XML router forwards XML documents to the appropriate application proxy. The XML router simply checks to make sure that a known public DTD is being used. If an unknown DTD is being used then the relevant security policy is applied. For example, the security policy for the detection of a malicious DTD could require the XML router to report the event to the security officer responsible for the management of the firewall. Within an XML document any embedded document type definition will take precedence over a DTD defined in the header of an XML document. Consequently an intruder may try to subvert the system by including an unauthorized DTD definition within an XML document. The XML router will therefore strip out and replace the DTD within an XML document. This ensures that a secure DTD is embedded within the XML document that is passed to the application proxy. The routing of XML documents to application proxies is achieved through the use of a simple database. Within the XML router a DTD is always associated with precisely one application. This association allows the router to pass a DTD to its appropriate application proxy. However, an application will be associated with a minimum of one DTD, it may be associated with many.

A DTD is also associated with precisely one incoming security policy. An incoming security policy may be associated with many DTDs. This incoming security policy defines the actions to be taken and the rules to be applied when an XML document is received from the HTTP proxy. For example, an incoming security policy may define the maximum size of a document is 4096 bytes and the minimum size is 1024 bytes. In addition the security policy may also define that violations are only to be reported to the security officer (i.e. no error messages are to be reported to the source of the XML document).

A DTD is also associated with precisely one outgoing security policy and an outgoing security policy may be associated with many DTDs. The outgoing security policy defines the actions to be taken and the rules to be applied when an application proxy receives an outgoing XML document.

Typical incoming and outgoing security policies for specific DTD will include:

- People from whom the system is allowed to accept XML documents encoded using a particular public DTD.
- People who are allowed to send XML documents encoded using a particular secure DTD.
- Maximum and minimum permitted size of an XML document.
- Comparison between the public DTD used in an incoming message and its secure version.
- How and under what conditions errors are reported to the source of the XML document.
- How and under what conditions errors are reported to the firewall and other security authorities.
- Security countermeasures to be employed when a security rule is repeatedly broken.

The routing for a particular DTD and the associated security policies are parameters of the XML firewall that are managed by the security officer.

4.4 XML Application Proxy

The XML Application Proxy has three main functions. The first is to validate incoming XML documents and document content. The second is to convert incoming documents from XML format to native Application format. The final function is to convert outgoing native Application format documents to XML format.

For an incoming message the public DTD will have been stripped by the XML Router and replaced with the equivalent secure DTD. The Application Proxy will also only receive documents sent with a DTD that can be handled by the Application Proxy. When dealing with an incoming message the proxy applies three processes:

- **XML Parser**. This makes use of a look-up table that indicates the form of parsing that is to be performed on different document types according to their DTD. Each Application Proxy will contain a single parser. The look-up table specifies different forms of parsing depending on the DTD. The parser takes as input an XML document and produces as output a DOM tree or a series of parsed events depending on the type of parser. In the event of an unsuccessful parse, the message will be discarded and security policies will be followed.
- **Application Engine**. The application engine contains a set of document specific rules that specify the way in which the DOM tree can be mapped into native Application format. These rules will be specialized to a specific document type within the Application. If the Application is XML enabled, it is possible that the output of the Application Engine may be an XML document.

The Proxy also identifies the SIP of the Application it serves. The XML Parser, Validator and Application Engine will communicate though standardized APIs. When dealing with an outgoing message the proxy applies a single process, the application engine. The application engine contains a set of message specific rules that specify the way in which the native Application format message can be wrapped in XML. The Proxy also strips the SIP from the message to the XML Router. Outgoing XML messages are not validated as it is assumed that the wrapping generates valid XML documents.

5 Conclusion

The XEON architecture provides a mechanism for secure communication between e-commerce applications over an unsecured network. XML is used to wrap application specific messages for network delivery via HTTP, SSL and HTTPS protocols. In addition to the standard XML validation, generic rules, application specific rules and security policies can be embedded in the system. It is also possible to extend the architecture to make use of emerging XML security standards such as XML signatures [9] and XML encryption [10].

This approach has a number of other benefits; it makes it possible to network enable non-networked applications and it provides mechanisms to support interoperability between diverse systems such as those found in an e-commerce environment. The architecture also offers the ability to provide an evolution path for legacy information systems. XEON has been successfully implemented in a prototype form. Current work focuses on developing generic languages for expressing application specific rules and security requirements.

Acknowledgements

This research has been supported by QinetiQ Ltd as part of a corporate research programme for UK MOD, under contract number CU009-5080 and was conducted in the HEFCW funded Internet Technologies Research Lab in the School of Computing University of Glamorgan.

References

[1] A. Blyth, D. Cunliffee, and I. Sutherland, "Security analysis of XML, XML usage and XML parsing," *Journal of Computers and Security*, vol. 22, no. 6, pp. 494-505, 2003.

[2] S. Brown, *Implement Virtual Private Networks*, McGraw-Hill Professional Publishing, 1999.

[3] M. Goodland, and C.Slater, *SSADM Version 4.0 - A Practical Approach*, McGraw Hill, 1995.

[4] G. Lawton, "XML specs duel over E-commerce," *Computer Journal*, vol. 32, no. 4, pp. 17-19, 1999.

[5] S. Northcutt, M. Cooper, M. Fearnow, and K. Frederick, *Intrusion Signatures and Analysis*, New Riders, 2001.

[6] D. J. Polivy, and R. Tamassia, "Authenticating distributed data using web services and XML signatures," *Proceedings of the 2002 ACM Workshop on XML security*, ACM Press, 2002.

[7] X. Wang, G. Lao, T. DeMartini, H. Reddy, M. Nguyen, and E. Valenzuela, "XrML - eXtensible rights markup language," *ACM Workshop on XML Security*, ACM Press, Nov. 22, 2002.

[8] WWW Consortium, *The XML Specification*, 2000. (http://www.w3c.org/XML)

[9] WWW Consortium, *XML-Signature Syntax and Processing*, 2000. (http://www.w3.org/TR/xmldsig-core)

[10] WWW Consortium, *XML Encryption Syntax and Processing*, 2000. (http://www.w3.org/TR/xmlenc-core)

[11] E. D. Zwicky, S. Cooper, and D. B. Chapman, *Building Internet Firewalls*, 2nd Edition, O'Reilly, 2000.

Andrew Blyth received his Ph.D. in Computer Science in 1995. He is currently Head of the Information Security Research Group (ISRG) at the Department of Electronics and Computer Systems Engineering, Faculty of Advanced Technology (FAT), University of Glamorgan, UK. He is published numerous papers in the area of computer network defense (CND). His research interests include: Computer Network Attack, Computer Network Defense and Intrusion Detections Systems.