THE IMPACT OF XML SECURITY STANDARDS ON MANAGING POST PROCESSED TELEMETRY DATA

Jeffrey R. Kalibjian
Hewlett Packard Corporation

ABSTRACT

Today many organizations use the Secure Sockets Layer protocol (SSL, now known as TLS, or Transport Layer Security) to secure post processed telemetry data transmitted over internal or external Internet Protocol (IP) networks. While TLS secures data traveling over a network, it does not protect data after it reaches its end point. In the Open Systems Interconnection (OSI) layer model, TLS falls several layers below the application category. This implies that applications utilizing data delivered by TLS have no way of evaluating whether data has been compromised before TLS encryption (from a source), or after TLS decryption (at the destination). This security "gap" can be addressed by adoption of a security infrastructure that allows security operations to be abstracted at an OSI application level.

KEY WORDS

Link data security, application data security, XML, TLS, XKMS, SAML, XACML, SOAP.

INTRODUCTION

The Extensible Markup Language (XML) is rapidly becoming the universal data exchange language. Additionally, the concepts of XML encryption and XML signatures have been established to provide privacy, integrity, and authorization in XML documents. These fundamental building blocks are being further used by both the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS) to define more sophisticated security abstractions utilizing XML including XKMS, the XML Key Management Specification; SAML, the Security Assertion Markup Language; XACML, the XML Access Control Markup Language; and, SOAP, the Simple Object Access Protocol. After providing overviews of these important application security standards, examples will be given of how they maybe used to better secure and manage telemetry post processed data.

APPLICATIONS AND NETWORKED COMPUTERS

With the connectivity of computers via networking came the goal of developing applications that would both enable individuals on those computers to exchange information and communicate as well as develop applications that could autonomously exchange information with each other to perform business operations [1]. It has turned out that the former goal has been much easier to achieve; examples of tools that have been developed to allow humans to communicate include e-mail (SMTP) and the World Wide Web (HTTP and HTML). Achieving the later goal has been fraught with promise and frustration. Concepts such as the Distributed Computing Environment (DCE) evolved that attempted to standardize various remote procedure call approaches to remote application interaction. The Common Object Request Broker Architecture (CORBA) [2] attempted to standardize distributed procedure technology based on making requests on remote objects. DCE never attained widespread industry support, while CORBA wasn't as scalable. While these technologies found their place in product offerings they never reached the ubiquity of use as the web (Hyper Text Transfer Protocol) or e-mail (Simple Mail Transfer Protocol).

INFORMATION EXCHANGE BASICS

In order to exchange information, there needs to be a paradigm for understanding the message and the data in the message as well as a method for sending it. The success of the WWW has made HTTP the communication mechanism of choice. Today most standards bodies involved with Internet protocol or process definition utilize the Extensible Markup Language (XML) for data definition.

THE HYPERTEXT TRANSFER PROTOCOL

HTTP [3] is a simple client/server stateless communication protocol that runs on top of the TCP/IP protocol. Its primary purpose is to have a server supply resources requested by a client. Typically the resources are files (e.g. HTML page), but they can also be dynamically generated data as well. The communication model is very simple. A client opens a connection (to port 80 of the server) and sends a request message to a HTTP server. A request has an initial line comprised of three parts separated by spaces: a method name, the local path of the requested resource, and the version of HTTP being used. The most common method is GET. In the example below the HTTP 1.0 protocol is being used to request the recipes file:

```
GET /home/data/recipes.html HTTP/1.0
```

A response is similar in that there is an initial line also comprised of three parts: the HTTP version, response code, and English "reason" phrase:

```
HTTP/1.0 200 OK
```

Actually after the initial line in both the request and response there can also be zero or more header lines, a blank line (carriage return, line feed), then an optional message body (in the response this is the requested data). A more complete example is shown below in Figure 1.

Request

```
GET /home/data/colors.html HTTP/1.0
User-Agent:  AHTTPBrowser/1.1
<CR><LF>
```

Response

```
HTTP/1.0 200 OK
Server: PiratedServer/1.1
<CR><LF>

Content-Type: text/html
Content-Length: 50
<html>
<body>
<h1>The Colors are:</h1>
<p>red <br>green<br>blue<p>
</body>
</html>
```

Figure 1. A simple HTTP request/response dialog. Observe the MIME information in the response message body.

THE EXTENSIBLE MARKUP LANGUAGE

The Extensible Markup Language (XML) [4][5] is what its name implies: a markup-language much like HTML. However, unlike HTML, XML was designed to describe data and explain what the data is. HTML focuses on displaying data and specifying how displayed data should be rendered on a screen. The central element of a XML file is its Document Type Definition (DTD) [6] that effectively defines the legal entities allowed to be in the file.

- Elements---Primary anchor entities.
- Tags----markup elements.
- Attributes—provide additional important information about elements.
- Entities---variables used to define common text.
- PCDATA—parsable character data.
- CDATA—character data that is not parsable.

Consider the following example describing rate data:

```
<?xml version="1.0"?>
<!DOCTYPE rate [
  <!ELEMENT rate (time, rate_x, rate_y, rate_z)>
  <!ELEMENT time          (#PCDATA)>
```

```
  <!ELEMENT rate_x        (#PCDATA)>
  <!ELEMENT rate_y        (#PCDATA)>
  <!ELEMENT rate_z        (#PCDATA)>
]>
```

An XML file containing such data might look like the following (assume that the above definition was in a local file named rate.dtd):

```
<?xml version="1.0"?>
<!DOCTYPE rate SYSTEM "rate.dtd">
<rate>
      <time>1.10</time>
      <rate_x> 04234563 </rate_x>
      <rate_y> 01232324 </rate_y>
      <rate_z> 02345589 </rate_z>
</rate>
```

The power of XML is in the DTD, this allows independent groups of people to interchange data in a standard way. An alternate to a DTD is something called an XML Schema.

PUTTING SOME OF THE PIECES TOGETHER

The Simple Object Access Protocol (SOAP) [7][8] is XML based and defines a message protocol that enables applications to exchange information over HTTP. Since SOAP is defined using XML it has many nice attributes including platform independence, language independence, and extensibility. In addition, utilizing HTTP for the transport mechanism enables SOAP messages to pass through firewalls and be handled appropriately by proxy servers (a claim RPC based CORBA can't make). Thus, applications written in different programming languages and running on different operating systems can utilize SOAP to communicate in a seamless manner. An outline of a SOAP message is illustrated in Figure 2. Observe that a SOAP message use the SOAP envelope (http://www.w3.org/2001/12/soap-envelope) and encoding (http://www.w3.org/2001/12/soap-encoding) namespaces.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
  ...
  ...
</soap:Header>
<soap:Body>
  ...
  ...
  <soap:Fault>
    ...
    ...
```

```
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

Figure 2. An outline of a SOAP message.

THE SECURITY FACTOR

An infrastructure to flexibly and extensibly define data, as well as a mechanism to send/receive messages containing such data, is a necessary requirement for remote application interaction. Yet, it is not sufficient. Sufficiency requires that the messages and data be secured. Specifically, that

- They can be kept private (privacy).
- Their originating application can be confirmed.
- Their integrity can be validated (i.e. that the message was not altered during transit).

There are two XML security standards that implement these security notions

- XML Digital Signature [9] provides authentication and integrity.
- XML Encryption [10] provides privacy.

The XML digital signature can be applied to an entire XML document, multiple XML documents, non-XML documents, specific elements in an XML document or element content portions of an XML document. XML signatures can even contain other XML signatures. The XML signature itself maybe stored separate from the data the signature is over (detached signature), or in the same XML file (enveloped signature). The primary signature elements include

- <Signature> -- The XML signature element.
- <SignatureValue> -- The cryptographic signature.
- <SignedInfo> -- The data that was signed. Will typically contain <Reference> elements that contain a digest of the data, digest algorithm and a URI to where the data can be found. URI's may refer to non-XML content. Reference elements may also specify transforms to be performed on data before they are digested. Canonicalization transforms exist that insure simple variations in syntax (e.g.whitespace) will not prevent "identical" texts from "digesting" differently.

The XML encryption facility, like the signature capability, can also be applied to an entire XML document, multiple XML documents, non-XML documents, specific elements in an XML document or element content portions. The primary encryption element is the <EncryptedData> element. This element will replace an element that is encrypted. The <EncryptedData> element may include a type attribute that describes the MIME type of the element or element content that was encrypted. The <EncryptedData> element also defines the algorithm that is used for encryption. User defined properties may also be associated with an encrypted element. Other encryption elements include

- <CipherData> -- Stores result of encryption operation.
- <CipherValue> -- Contains actual encrypted data.
- <CipherReference> --- URI reference to encrypted data stored at another location.

Thus, we see in the encryption and digital signature realm, XML security delivers true end-to-end security while facilitating selective privacy. This is in direct contrast to security protocols like TLS that provides only point-to-point security at non-application layers and does not provide for a selective encryption capability. These are key shortfalls because it precludes TLS from being readily utilized in a general computer networked environment in which automated (business) applications are exchanging information. This is because in these situations many intermediate applications need to process pieces of a transaction. For these scenarios only XML type security will suffice.

OTHER IMPORTANT XML STANDARDS

Key management, authorization, and authentication must be robust in a secure application environment. There are three XML based initiatives supporting these areas:

XML Key Management Specification [11] (XKMS) – This specifies the XML message formats (implemented with SOAP) necessary to support interaction for public key management including certificate request/response, certificate revocation, identity query, etc. It includes a key registration service specification and a key information service specification.

Security Assertion Markup Language [12] (SAML) – This specifies a mechanism to share authorization and authentication information (assertions). This data can be shared via a request/response messaging format implemented with SOAP. Possible applications include single sign on.

XML Access Control Markup Language [13] (XACML) – This specifies policies for making authorization decisions (i.e. whether to grant a privilege/action given an entity has been authenticated, etc.)

THE WEB SERVICE ABSTRACTION

A "web service" [14] is defined to be computer-to-computer application interactions within or between enterprises utilizing platform independence and programming language neutrality concepts. The web service model uses XML, HTTP and SOAP to link applications together. A key requirement of web services is that they be secure. There are seven proposed Web Service specifications to achieve this

WS-Security [15] – Specifies how XML signatures and XML encryption maybe used to secure Web service.

WS-Policy – Specifies how security policies impact intermediate entities and end points.

WS-Trust – Specifies how trust can be established between web services.

WS-Privacy – Specifies how web services and web service clients can define and establish privacy preferences.

WS-SecureConversation – Specifies how to dynamically establish trust using key exchange.

WS-Authorization – Specifies how authorization can be managed in a web services.

WS-Federation -- Specifies trust management in a heterogeneous federated environment.

XML AND TELEMETRY DATA: NEW OPPORTUNITIES

While CCSDS [16] recommended packet formats define protocols for passing data from one device to another, the Spacecraft Markup Language (SML) [17] was released in late 1999 in an attempt to establish a neutral and extensive approach for describing contents of exchanged data packets. While there have been many attempts to develop languages to do this in the past (e.g. Record Definition Language, RDL) they all suffered from some drawbacks. SML hopes to leverage the tremendous infrastructure being developed for XML (e.g. compilers, etc) and make it available to the telemetry community. Opportunities also abound for XML and associated XML security standards to significantly impact telemetry data reduction and analysis. The following areas can be affected:

- Data Sharing – Reduced data maybe placed in XML formatted files. By making the DTL or XML schema for classes of data freely available to partner organizations data maybe easily exchanged. Each partner company would also be free to develop their own custom applications that could further reduce/analyze the agreed upon data DTL. The XML files could be moved utilizing SOAP messages.

- Data Security – XML Encryption could be used to secure telemetry data in XML files and while the data is in transit. This would result in complete end-to-end privacy for the telemetry data in contrast to systems that might encrypt the data while in transit, but leave the data unencrypted when it reached its endpoint destination. In addition, since the

encryption granularity is at an element level, the same file could contain both unencrypted (when appropriate) and encrypted data.

- Remote Group Analysis – The DTL of the telemetry data could be expanded to include XML digital signatures that could, in turn, be utilized to indicate who has reviewed the data. As the data is passed around to the appropriate parties, each party can see who has reviewed it. The DTL could even be modified to contain reviewer comments (which could also be digitally signed and encrypted); so as different parties reviewed the data, they could review the comments and be aware of the concerns of the other data reviewers.

- Automated Analysis – Applications could be written that could use the DTL or Schema to make sure data had been put through an appropriate business flow. That is, an application could check to make sure that the data had been seen/reviewed/approved/disapproved by appropriate engineering or QA personnel. If problems with the analysis are detected (i.e. digital signatures not obtained, or signatures indicating concern with data) automated alerts could sent to management for further action.

- Rule Enforcer – Once data is in the appropriate XML format, it may undergo required automated processing before engineers are allowed to examine the data. An example of such automated processing might be calibration.

CONCLUSION

The Hypertext Transfer Protocol together with the Extensible Markup Language are providing the foundation for a platform neutral approach for connected applications to exchange information. The Secure Object Access Protocol and XML Signatures/Encryption are providing the practical framework by which data can be seamlessly and securely interchanged. These technologies can and will have an impact in telemetry data analysis and have specific application in the areas of data sharing, data security, remote group analysis, automated analysis and rule enforcement. Even more benefits to telemetry post processing and analysis lie ahead when a robust Web Services framework (based on SOAP, XML, XML Encryption/Signature) is architected by the standards community.

References

[1] Oqbuji, Uche, “The History of Web Services, Parts 1 and 2,” September 2002, http://www.webservices.org.

[2] Object Management Group,”Common Object Request Broker Architecture (CORBA/IIOP), Version 3.0,” Object Management Group, December 2002, http://www.omg.org.

[3] Fileding, R; Gettys, J; Mogul, J; Frystyk, H; Masinter, L; Leach, P; Berners-Lee, T, “Hypertext Transfer Protocol-HTTP/1.1,” World Wide Web Consortium (W3C), June 1999, http://www.w3.org.

[4] Bary, Tim; Paol, Jean; Sperberg-McQueen, C.M.; Maler, Eve, “Extenisble Markup Language (XML) 1.0 Second Edition,” World Wide Web Consortium (W3C), October 2000, http://www.w3.org.

[5] W3 Schools, “XML Tutorial”, 1999-2003, http://www.w3schools.com/default.asp.

[6] W3 Schools, “DTD Tutorial”, 1999-2003, http://www.w3schools.com/default.asp.

[7] Box, Don; Ehnebuske, David; Kakivaya, Gopal; Layman, Andrew; Mendelsohn, Noah; Nielsen, Hennik; Thatte, Satish; Winer, Dave, “Simple Object Access Protocol (SOAP) 1.1,” World Wide Web Consortium (W3C), May 2000, http://www.w3.org.

[8] W3 Schools,”SOAP Tutorial”, 1999-2003, http://www.w3schools.com/default.asp.

[9] Bartel, Mark; Boyer, John; Fox, Barb; LaMacchia, Brian; Simon, Ed; “XML-Signature Syntax and Processing,” World Wide Web Consortium (W3C), December 2002, http://www.w3.org.

[10] Imamura, Takeshi; Dillaway, Blair; Simon, Ed, “XML Encryption Syntax and Processing,” World Wide Web Consortium (W3C), December 2002, http://www.w3.org.

[11] Ford, Warwick; Hallam-Baker, Phillip; Fox, Barbara; Dillaway, Blair; LaMacchia, Brian; Epstein, Jeremy; Lapp, Joe,” XML Key Management Specification (XKMS),” World Wide Web Consortium (W3C), March 2001, http://www.w3.org.

[12] Farrell, Stephen; Reid, Irving; Orchard, David, et. al., “Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML),” Organization for the Advancement of Structured Information Standards (OASIS), November 2002, http://http://www.oasis-open.org.

References (continued)

[13] Organization for the Advancement of Structured Information Standards, "Access Control Markup Language (XACML)," Organization for the Advancement of Structured Information Standards (OASIS), February 2003, http://http://www.oasis-open.org.

[14] The IBM Corporation and the Microsoft Corporation, "Security in a Web Services World: A proposed Architecture and Roadmap," IBM Corporation, Microsoft Corporation, April 2002, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/securitywhitepaper.asp.

[15] Atkinson, Bob; Della-Libera, Giovanni; Hondo, Maryann; et. al., "Web Services Security (WS-Security)," IBM Corporation, April 2002, http://www-106.ibm.com/developerworks/library/ws-secure.

[16] Consultative Committee for Space Data Systems, "CCSDS Charter," Consultative Committee for Space Data Systems, May 1999, http://www.ccsds.org/about/charter.html.

[17] Cappelaere, Jay; Offutt, Jay; "Spacecraft Markup Language, Draft Specification," Interface and Control Systems, Inc., December 1999, http://www.interfacecontrol.com/sml/Sml/SML_MMM.1.5.pdf.