Secured Object Signatures for Web Objects and XML Objects

Chang, H., Peter

Abstract— ***This paper presents object-oriented security models for a secured object signature of web and XML objects. The models address web and XML securities in general. They address in particular security wrappers for web pages, security nodes for network machines, and mid-tier securities for middle-tier machines. This paper discusses in a high level implementations for security nodes and mid-tier securities. It discusses an implementation for security wrappers in detail. The implementation encrypts web and XML objects in a company's Intranet web site and enhances the protections of them against unauthorized accesses by an intruder. The encryption method uses the combination of a randomly generated session key, a session id that consists of remote host id and port number, and an access sequence number that is maintained by the web server. Although the encryption method addresses the security wrappers, its approach is equally applicable to security nodes and mid-tier securities as well.***

Index Terms—*object-oriented, security, Web, XML, XSL*

1. *Introduction*

As Internet e-commerce activities grow more and more every year, Internet security is also becoming more important in securing the capitals invested in these activities. Oppliger [8] stated in a workshop held by the Internet Architecture Board (IAB) back in 1994 that scaling and security were considered to be the two most important problem areas for the Internet as a whole. Chang [4] uses an object-oriented analysis and modeling approach to define the architecture of a proposed method for securing web objects. It is known that object has both attributes and behaviors. Blaha et al. [2] covers in detail the object-oriented terminology and technology. Network machines, web servers, web documents (XHTML web pages), XML documents, XSL documents, database servers, and mid-tier machines are examples of web or XML objects.

For web documents, examples of attributes are URL web address, title, and hyperlinks. Examples of behaviors are navigation to other web pages through hyperlinks and the dynamic operations implemented by JavaScript functions defined in the web pages.

This paper covers the security model for web and XML documents, network machines, and mid-tier machines. For brevity, it concentrates on web and XML documents and covers the network machines and mid-tier machines in high level. The implementation approach is applicable to the latter two. Web pages and XML documents are stored or generated by programs on a web server. They may be protected from unauthorized people with the addition of a security mechanism in the server programs. Static web pages in XHTML files are in general accessed through hyperlinks in other web pages. Static web pages can also be represented in XML and XSL files. XHTML files, XML/XSL files, and XML files in general are more vulnerable to unauthorized accesses by an outside intruder. This paper concentrates the discussion on the intranet web site as it is used more for B2B activities.

Common encryption techniques involve with encrypting the whole document. Kalakota et al. [6] contains general discussions on encryption and web transaction security. W3C [12] provides general guidelines on securing XML documents for Web services. W3C [13] provides guidelines for XML digital signatures. Encryption processes that encrypt whole documents generally incur performance overheads. Sometimes, it is not the whole content of the documents that need to be encrypted. Rather it is how the access to these documents, for example, a hyperlink for a web document, needs to be protected through encryption.

This paper presents a selective encryption approach in the sense that instead of encrypting a whole document it encrypts a selected element of the document, using the hyperlink element as an example. In other words, it uses an encryption method that protects the access to XHTML documents or their XML/XSL equivalences through hyperlinks. This paper names the encrypted result for a hyperlink a *secured object signature* of the hyperlink. The session key, which is used as the encryption key, is first generated when a user logs into the web site. This paper introduces a concept of

Manuscript received October 12, 2005. P. H. Chang is with the College of Management, Lawrence Technological University, Southfield, Michigan, U.S.A. (e-mail: chang@ltu.edu).

including placeholders and references to templates to the hyperlinks. It introduces the access sequence number that tracks a user's access, which is incremented each time the user of the current session accesses a XHTML document or its XML/XAL equivalence through a hyperlink. This paper also introduces the algorithm for a generic Common Gateway Interface (CGI) script, implemented in the Perl language, that handles the events of the access through hyperlinks. This paper uses a database managed by a Relational Database Management System (RDBMS) to track and update the access sequence number. Hoffer et al. [7] covers the RDBMS technology. This paper presents one concrete example for the implementation that an XHTML document or its XML/XSL equivalence for a company's phone directory is listed through a hyperlink on the company's intranet home page. In this paper, after the encryption method is applied to the hyperlink reference, two scenarios will be discussed. The first scenario is that a valid user accesses the phone directory through the hyperlink and gets through. The second scenario is that an outside intruder attempts to access the phone directory and is denied the access.

2. SECURITY ARCHITECTURE FOR WEB OBJECTS

Figure 1 describes an object model for a client machine that hosts a web browser application, which accesses web or XML documents on web servers in a company's intranet or the Internet. The rest of this paper considers a web page or an XML document as an object instance of the WebDocument object class. For an ease of discussion, this paper concentrates the discussion on web pages up to Section 5. Discussions on XML documents will follow on Section 6.

When the browser requests a web page from a web server, the web server downloads the web page to the client machine to be rendered by the browser and displayed to the user.

In object-oriented terminology, a relationship among object classes is called an association. Binary association, the association between two classes, is the most common ones. In some cases, an object class may be associated with itself due to that it has two roles, for example, a person class that models a company's workers may have two roles: manager and employee. Such an association is called the unary association.

Figure 1 shows examples of binary associations. One of them is the association between the WebServer class and the WebDocument class. Figure 1 also shows examples of unary associations. One of them is the association between web pages that contain instances of the XHTML anchor tag for hyperlink: <a href="url">, referred as the link source, and the web pages identified by the web address' url, referred as the link target. For each web page as an instance of the WebDocument class, there may be zero or more hyperlinks contained therein. Each web page may also be referred to as link reference by zero or more web pages. Figure 1 shows the multiplicity reflecting the many-to-many association. For a many-to-many association, it is necessary to create an association class to correctly model the association because there may be attributes that belong more appropriately to the association rather than to the WebDocument class. The association class is depicted as SecurityWrapper in Figure 1. This association class may model security methods such as encryption and database management system (DBMS) facilities for the web pages. Blaha et al. [2] discusses in detail the association class.

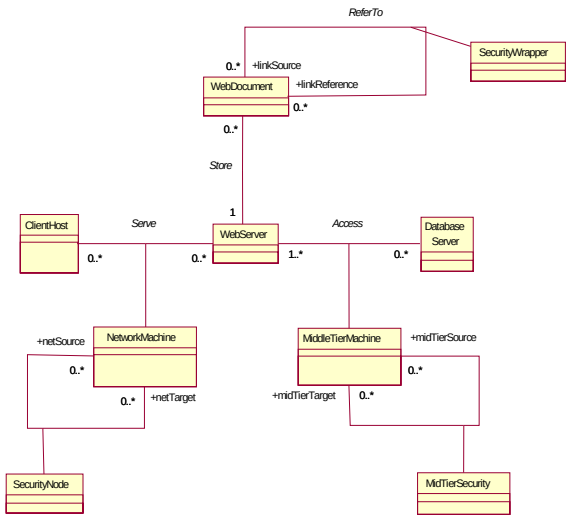

Figure 1. Object Model for the Security Architecture

The communication protocol between the client machine and the web server is HTTP. Data packets are transmitted between the client and server through network machines, which are modeled as the object class NetworkMachine in Figure 1. Examples of the network machines are hubs, switches, routers, domain name servers, and so on. A network machine may transmit data packets to many other network machines. Conversely a network machine may also receive data packets transmitted from many other network machines. So this association is a many to many unary association. From the same reason stated above, it is necessary to create an association class for this many-to-many association. This association class may model a security node that enforces security for data transmission depicted as SecurityNode in Figure

1. Examples of security nodes are firewalls and proxy servers.

Web servers may also need to access database servers through intermediate machines that host middleware software, which are modeled as the object class MiddleTierMachine in Figure 1. Similarly there is a many-to-many unary association on the MiddleTierMachine object class because a mid-tier machine may transmit (receive) data packets to (from) many other mid-tier machines. It is also necessary to create an association class for this association. This association class may model the security among mid-tier machines depicted as MidTierSecurity in Figure 1.

The rest of this paper generally refers the three association classes: SecurityWrapper, SecurityNode , and MidTierSecurity as the security association class. Chang [4] presented an earlier version of the object model that addresses the association class SecurityWrapper, but does not address the association classes SecurityNode and MidTierSecurity.

3. IMPLEMENTATION OF SECURITY ASSOCIATION CLASS

A security standard such as ISO X. 700 generally addresses security in areas that contain access, authentication, authorization, and auditing. Additionally areas like data confidentiality, data integrity, and non-repudiation services to communicating peers are also emphasized. In addition, some object-oriented approaches for security modeling addresses the issues of security in each of the seven layers of ISO's OSI model.

Many security implementations nowadays use configuration parameters stored in files instead of an operational databases because the number of these parameters is generally smaller than the number of transactional records that require a database and it is generally simpler to add, update, and delete the parameters values in files. Redundancy, however, leads to inconsistency. For example, suppose a security parameter value is stored in multiple files. In the event that this value needs to be changed, it should be changed in all the files that contain it. If it is not changed in one file, then the security measure related to this parameter is compromised. The object model depicted in Figure 1 provides a solid foundation for implementation to databases. The benefit of database such as a relational database is that the redundancy of data is minimized. The performances of DBMS' and the server software and hardware platforms that support them are getting better. In addition, the DBMS itself is a security layer that adds extra level of security that the file system is lack of. For example, most relational database management systems (RDBMS') provide record and field level security, while file system provides only security at the file level.

Implementations of the *SecurityNode* association class include firewalls, screen routers, and proxy servers, which use the approach of packet filtering to examine and control the transmission of data packets on the exterior network entering the intranet. Oppliger [8] covers firewalls in more details. Oppiger's paper, however, does not address how are the security configuration data such as values of security parameters stored. The object model in Figure 1 may be expanded as a basis to design databases that store them.

Implementations of the *MidTierSecurity* association class include establishing and enforcing security policies for applications that span multiple administrative domains. The policy addresses among others security events and the associated services. A commonly adopted paradigm for event services is the publisher-subscriber model. Middleware software resides on the mid-tier machines. Tripathi [10] covers the middleware security. Tripathi's paper also does not address how are the security configuration data such as values of security parameters stored. The object model in Figure 1 may also be expanded as a basis to design databases that store them.

As for an implementation of the *SecurityWrapper* association class this paper proposes a method of using encryption and database managed by a DBMS. Common Gateway Interface (CGI) script is one protocol that a web server uses to respond to clients' requests sent by web browsers through web pages. Other protocols include Microsoft's Active Server Page (ASP) and Sun's Java Server Page (JSP) and servlets. These other means are aligned with web services; for example, ASP is with Microsoft ASP.NET (Tabor [9]), and JSP/Servlet is with Sun Sunone (Watson [11].) Web services are based on XML, which will be discussed in Section 6.

For brevity this paper assumes that the web servers execute CGI programs responding to clients' requests. CGI is usually implemented with programming languages such as Perl or C++/C. These languages require relatively less supporting libraries than those languages used for J2EE or .NET framework. The CGI examples presented in this paper were developed in the Perl language (Bunce et. al. [1]). The same method is applicable to other protocols and programming languages such as Java or C#.

Such protocols and tools are parts of the implementations of the middleware concept. Chang [3] discusses similarities among implementations of the middleware.

The association class *SecurityWrapper* may model any security feature that secures the implementation of the association between the group of instances of WebDocuments with link source role and the group of instances of WebDocuments with link target role. This paper proposes the addition of an encryption wrapper to the URL address in the XHTML anchor tag as the security feature. The encryption is implemented in CGI programs written in the Perl language. The key for the encryption is generated randomly. After its generation, its value is maintained in a database table. The remaining sections of this paper describe this implementation.

4. SESSION ID

There are in general two types of access to web documents. The first type is a viewing of static web page; for example, reading a newspaper article on one of the web site of a newspaper company. In this case, a reader simply browses over the static web pages. The web server does not need to know the identification of the reader and hence does not need to maintain a session with the reader. The second type is transaction oriented, that is, a user interacts with the web page by providing input data for a period of time; for example, doing online shopping through an implementation of the shopping cart design. In this case, a user needs to provide input to multiple web pages. The web server needs to know an identification of the user and hence needs to maintain a session with the user.

A web server maintains two environment variables associated with the client: REMOTE_HOST and REMOTE_PORT. The former contains the IP address of the client machine and the latter contains the port number used by the client machine to communicate with the web server.

Assume that a user uses a desktop or laptop personal computer to access web documents on a web server. The combination of REMOTE_HOST, REMOTE_PORT, and the web server system time at the first instance of access uniquely identifies the session for the user. From now on this paper considers a session id to be of the format: ipAddress-portNumber-timestamp, where ipAddress is the value of REMOTE_HOST, portNumber is the value of REMOTE_PORT, and timestamp is the system time on the web server. This paper addresses an approach that when a user logs on to a web page to start a session, the login CGI script generates a session id as was just described and a session key to be used as the encryption key.

5. PLACEHOLDERS AND ENCRYPTION

In many cases, intranet users attempt to log into their intranet web sites from remote locations outside their working facilities through the Internet. In responding to a client request for a user's login, after authenticating the user, a CGI script generates a home web page containing welcoming information, which may possibly include hyperlinks.

A scenario is that the home web page named, say, index.html in a company's intranet web site includes a hyperlink to the employee phone directory as shown in Figure 2.

```
<a href="phoneDirectory.html">Phone Directory</a>
```

Figure 2. Hyperlink to the web page for Phone Directory

The phone directory may be a static XHTML file stored on the web server. Thus, if a user successfully logs on the company's intranet web site, the user can freely access the phone directory through the hyperlink. Inductively, the user can freely access any static XHTML file that is referenced by a hyperlink on the web page that the user is currently on. As was mentioned in Section 4, the IP address of the client machine and the port number used by the client machine are contained in the two environment variables: REMOTE_HOST and REMOTE_PORT, respectively. If the values for these two variables, which form the session id, are intercepted on the Internet by an intruder, the intruder can conceivably at least get onto the company's intranet web site and access all static XHTML files that are referenced by hyperlinks in the intranet home page.

```
<a href="getcontent.pl?template=phoneDirTemplate.html
&sessionId=ZZZZZZS&sessionKey=ZZZZZK
&accessSeqNumber=ZZZZZA">Phone Directory</a>
```

Figure 3. Hyperlink in index.html with Placeholders

This paper presents an encryption algorithm that adds an extra layer of security that protects an important Web object such as the hyperlink in Figure 2. An implementation code for this algorithm is available from the author upon request. From this code, the decryption algorithm can be derived. This paper's goal is using an algorithm that is effective, but not very complex and thus does not add too much overhead to the processing time.

This paper first proposes to replace the

reference to the name of the static web page in the home web page index.html by the name getcontent.pl of a CGI script with a template file name and placeholders as its arguments as shown in Figure 3.

The template file phoneDirTemplate.html is a template file for the static web page phoneDirectory.html discussed previously. It differs from the latter only in that it contains placeholders as replacements for static links in the latter. These static links are inserted by the owner of the file phoneDirectory.html that allow a user to navigate to some common web pages such as the home page. Figures 4 shows a hyperlink in phoneDirectory.html to home web page. Figure 5 shows the revised hyperlink in the template file phoneDirTemplate.html.

```
<a href="index.html">Home</a>
```

Figure 4. Hyperlink in *phoneDirectory.html* to the web page for home

```
<a href="getcontent.pl?template=../htdocs/index.html
&sessionId=ZZZZZZS&sessionKey=ZZZZZK
&accessSeqNumber=ZZZZZA">Home</a>
```

Figure 5. Hyperlink in *phoneDirTemplate.html* with Placeholders

Assume that the login web page contains the following line:

```
<form method="post" action="login.pl">
```

where login.pl is the name of the CGI script that handles the authentication of users. The login.pl script sets up a session key, a two digits number less than 62, as the key for the encryption. The reason that the session id is not used as the key is that it tends to be a long string, which is not suitable for the encryption algorithm that is propose in this paper. The key steps of the login.pl script involve with setting up encryption environment and session id, authenticating the user who logs in, updating database table with 0 as the initial access sequence number, encrypting session id and the access sequence number, and replacing the placeholders ZZZZZZS, ZZZZZK, and ZZZZZA in Figure 3 by the encrypted data for session id, the session key, and the encrypted data for access number, respectively. The resulting hyperlink acts like a secured object signature for the hyperlink in Figure 2.

To show that the encryption secures the static hyperlinks, this paper considers two scenarios. The first scenario is that a valid user logs on to the intranet home page and at some point of the session accesses the phone directory through its hyperlink. The second scenario is that an intruder intercepts a copy of a transmitted XHTML file and manages to get on the intranet home page. The intruder then attempts to access the phone directory through the hyperlink.

For the first scenario, a CGI script named getContent.pl first validates that the access sequence number matches the one in the database. If it does not, then it logs an error message and terminates the CGI program. Otherwise it updates the access sequence number in the database and displays the content of the template of the phone directory with the substitutions of encrypted session id, session key, and encrypted access sequence number for the placeholders in the hyperlinks, if they exist.

The key steps of the getContent.pl script involve with decryption and checking whether the access sequence number is in sequence. That is, whether it matches the access sequence number in the database. If they do not match, then there is a possible intrusion. In this case, reject the access. If they match, then increment the access sequence number in the database by 1, re-encrypt the this access sequence number along with session id, and repeat the steps of replacing placeholders as those for the login.pl script.

For the second scenario, the intruder intercepts a copy of the transmitted XHTML file. When the intruder gets on the company's intranet web site, the access sequence number is out of sequence, because the session for the valid user has already updated it in the UserAcess table in the database. If the intruder wants to access the phone directory through the hyperlink, the getContent.pl script will discover that the access sequence number with the session id and session key on the intercepted copy does not match the one in the UserAccess table in the database. The getContent.pl script then denies the intruder from accessing the company's phone directory.

6. *Encryption in XML/XSL and Selective Encryption*

XML is used in many areas nowadays such as Web services technology and databases. In particular, a web page in XHTML format can be represented by an XML file and an XSL file. For example, the XHTML element in Figure 2:

```
<a href="phoneDirectory.html">Phone Directory</a>
```

can be represented by the XML element:

```
<PhoneDirectory>
  <name>Phone Directory</name>
  <link>phoneDirectory.html</link>
</PhoneDirectory>
```

and the corresponding XSL element:

```
<xsl:variable name ="linkname">
    <xsl:value-of select = "PhoneDirectory/link"/>
</xsl:variable>
 <a href="{$linkname}"><xsl:value-of select =
 "phoneDirectory/name"/></a>
```

W3C [14] defines the XSL syntax. The XHTML element in Figure 3

```
<a href="getcontent.pl?template=phoneDirTemplate.html
   &sessionId=ZZZZZZS&sessionKey=ZZZZZK
   &accessSeqNumber=ZZZZZA">Phone Directory</a>
```

can also be represented by the following XML elements:

```
<PhoneDirectory>
  <name>Phone Directory</name>
  <link>getcontent.pl</link>
  <template>phoneDirTemplate.html</template>
  <sessionId>ZZZZZZS</sessionId>
  <sessionKey>ZZZZZZK</sessionKey>
  <seqNumber >ZZZZZA</ seqNumber >
</PhoneDirectory>
```

and the corresponding XSL elements:

```
<xsl:variable name ="linkname">
  <xsl:value-of select = "PhoneDirectory/link"/>
</xsl:variable>
<xsl:variable name ="template">
  <xsl:value-of select = "PhoneDirectory/ template"/>
</xsl:variable>
<xsl:variable name ="sessionId">
  <xsl:value-of select = "PhoneDirectory/sessionId"/>
</xsl:variable>
<xsl:variable name ="sessionKey">
  <xsl:value-of select = "PhoneDirectory/sessionKey"/>
</xsl:variable>
<xsl:variable name ="seqNumber">
  <xsl:value-of select = "PhoneDirectory/seqNumber"/>
</xsl:variable>
<a
href="{$linkname}"?template={$template}&sessionId={$s
essionId}&sessionKey={$sessionKey}&accessSeqNumber=
{$seqNumer}><xsl:value-of select =
"phoneDirectory/name"/></a>
```

XML is used in many other areas as well such as second-generation Web services and Service-Oriented Architecture (SOA). The role of a computer program as an XML parser is that it parses the structures of an XML data packet or an XML file. In the last example, the CGI script getContent.pl and the web browser are the XML parsers. The approach of using a template file with placeholders and applying an encryption method to convert the lines containing the placeholders to secured object signatures for certain XML elements can be applied to XML in general.

Geer [5] points out a research done by Ron Schmelzer of the market research firm ZapThink that XML will rise from 3 percent of global network traffic in 2003 to at least 40 percent by 2008. Some of the XML traffic may include overheads of full data encryption. The selective encryption approach proposed in this paper may be used with the binary XML approach discussed in Geer [5] to lessen the XML traffic volumes.

7. CONCLUSION

Ensuring web security is a complex problem that involves almost every aspect of the enterprise of an organization. Object-oriented analysis and modeling technique is especially adaptable to analyze complex problems and model solutions for them. This paper covered the secured object signatures based on a security model for web documents and XML documents, network machines, and mid-tier machines by using the object-oriented technique. The security model is based on three security association classes: *SecurityWrapper*, *SecurityNode*, and *MidTierSecurity* as described in Section 3. Although this paper concentrated on the security association class *SecurityWrapper* by presenting the approach that uses encryption methods and databases, the approach is applicable to the two other security association classes, which span the whole spectrum of the network infrastructure for web and XML applications.

REFERENCES

[1] Bunce, T., Descartes, A., "Programming the Perl DBI," *O'Reilly and Associates, Inc*., 2000.

[2] Blaha, M., W. Premerlani, W., "Object-Oriented Modeling and Design for Database Applications," *Prentice Hall*, 1998.

[3] Chang, P., "A Platform Independent Middleware Architecture for Accessing Databases on a Database Server on the Web," *IEEE*, IEEE R4 Conference, Indianapolis, Indiana, June 2003.

[4] Chang, P., "On An Architecture for Securing Web Objects," ASEE/NCS 2004 Conference, Western Michigan University, Kalamazoo, Michigan, April 2004

[5] Geer, D, "Will Binary XML Speed Network Traffic?," *IEEE*, IEEE Computer, April 2005, 16-18.

[6] Kalakota, R., Whinston, A., "Electronic Commerce, A Manager's Guide," *Addison Wesley Longman, Inc*., 1997.

[7] Hoffer, J., Prescott, M., McFadden, F., "Modern Database Management," 6th ed., *Prentice Hall*, 2002.

[8] Oppliger, R., "Internet Security: Fire Walls and Beyond.," *ACM*, Communications of the ACM, Volume 40 Issue 5, May 1997. 92-102.

[9] Tabor, R., "Microsoft .NET XML Web Services," *SAMS*, 2002.

[10] Tripathi, A., "Challenges Designing Next-Generation Middleware Systems," *ACM*, Communications of the ACM, Volume 45 Issue 6, June 2002. 39-42.

[11] Watson, M., "Sun ONE Services," 1st ed., *John Wiley & Sons*, 2002.

[12] W3C, "Web Service Architecture Requirements," http://www.w3.org/TR/2002/WD-wsa-reqs-20021114.

[13] W3C, "XML Signature WG," http://www.w3.org/Signature/

[14] W3C, "XSL Transformations," http://www.w3.org/TR/xslt, (XSLT), Version 1.0, 1999.