PART IV

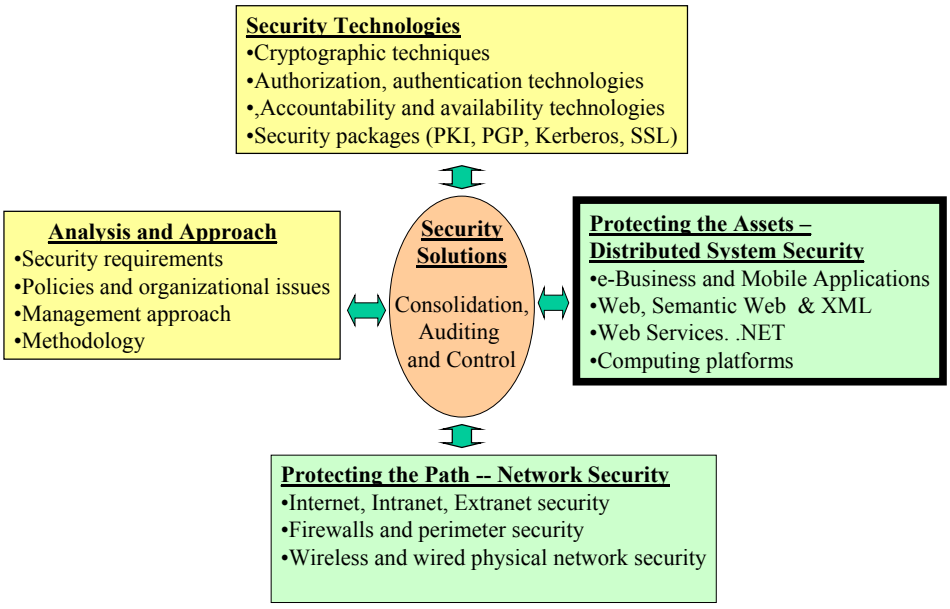
Protecting the Sites – Distributed System Security

This part of the book elaborates on securing the hosts and the databases, applications, and middleware services residing on the hosts (box with dark borders in the framework shown below).

Chapter 10: Web, Semantic Web, and XML Security

Chapter 11: Modern Distributed Platform, Web Services and .NET Security

Chapter 12: Application Security: Protecting e-Commerce and Mobile Applications



10 Web, Semantic Web, and XML Security

10.1 Introduction

Virtually all business, most government agencies, almost all educational institutions, and a very large number of individuals increasingly use the World Wide Web (WWW) on a regular basis. Due to this heavy reliance and the potential impact of Web security breaches, the need for a high level of Web security is obvious. This chapter discusses the various approaches and technologies being used for secure Web communications.

While the traditional first-generation Web technologies are deeply entrenched into our daily business and personal lives, several new technologies are being introduced to make the Web more powerful. This work, under the general umbrella of Next Generation Web and commonly known as the Semantic Web, has introduced powerful new technologies such as XML. However, Semantic Web has also introduced new security issues that are not associated with the classical first-generation Web. A great deal of attention to the Semantic Web security is paid in this chapter to highlight the new security issues.

Chapter Highlights

- Traditional Web security at present is widely supported through SSL.
- Cookies raise some privacy issues. They can be turned off but then you may not be able to use some of the sites that require cookies.
- Web proxies can be used to provide common security across multiple servers.
- Semantic Web raises new security issues that are not present in the traditional Web.
- XML security is somewhat unique because different segments of the same document may need different signers and different privacy considerations.
- XML signatures and XML encryption are specifically designed to address the special needs of XML security.
- Platform for Privacy Preferences (P3P) and PICS (Platform for Internet Content Selection) are major W3C initiatives to standardize privacy and content specifications.

10.2 Traditional Web Security and SSL

10.2.1 General Web Security Considerations

To develop an understanding of Web security, we first need to quickly review the Web building blocks. Fortunately, the Web is based on the following few simple concepts and technologies shown in Figure 10-1 (we reviewed these in Chapter 7):

- ***Websites*** provide the content that is accessed by Web users.
- ***Web browsers*** are the clients that support HTML for Web surfing.
- ***Uniform Resource Locator (URL)*** is the basis for locating resources in WWW.
- ***Hypertext Markup Language (HTML)*** is an easy-to-use language that tags the text files for display on Web browsers.
- ***Hypertext Transfer Protocol (HTTP***) is an application-level protocol designed for Web users.
- ***Web navigation and search services*** are used to search and surf the vast resources available in "cyberspace."
- ***Gateways to non-Web resources*** are used to bridge the gap between Web browsers and the corporate applications and databases.

A good approach for categorizing Web security is to think in terms of Web clients, Web servers, and the paths in between. The main issue with Web clients is the active content that appears as helper applications, Java applets, ActiveX control applets, and CORBA clients. Since this content is mostly application software, we will discuss it in the applications security chapter (Chapter 12). The main challenges in Web security are in the Web server area because Web servers are custodians of the HTML content, XML content, and Web gateways that provide access to back-end applications. These issues are also application-related and will be discussed in a later chapter. When the data travels

between the Web clients and servers, it needs to be protected. Secure communications, ensuring data privacy and data integrity, were discussed in Part III of this book.

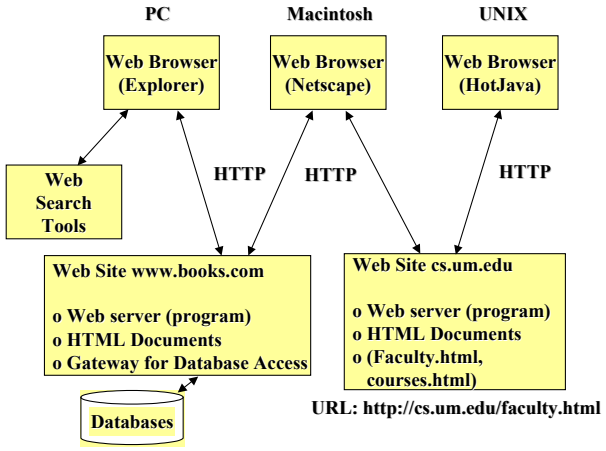


Figure 10-1: Conceptual View of World Wide Web

There are different approaches to achieving Web security. Figure 10-2 shows the main approaches. If you use the network-level security by employing IPSec (Figure 10-2a), then it is transparent to all applications. Since Web runs on top of HTML, all communications between Web clients and servers are protected by using this option without any changes to Web browsers and servers. However, IPSec requires a re-engineering at the router level. Another option is to move security to a higher level, i.e., above the TCP layer as shown in Figure 10-2b. The most widely used option is SSL (Secure Socket Layer). SSL could be made transparent to the applications by using it as part of the TCP/IP stack, or it can be packaged in specific applications. For example, SSL is currently packaged with Web browsers and servers. Application-specific security services such as PGP, S/MIME, and SET (Secure Electronic Transactions), shown in Figure 10-2, are used to protect specific applications. These application-specific security packages are discussed in a later chapter. Due to the heavy use of SSL in Web security, we will take a closer look at SSL in Section 10.2.2.

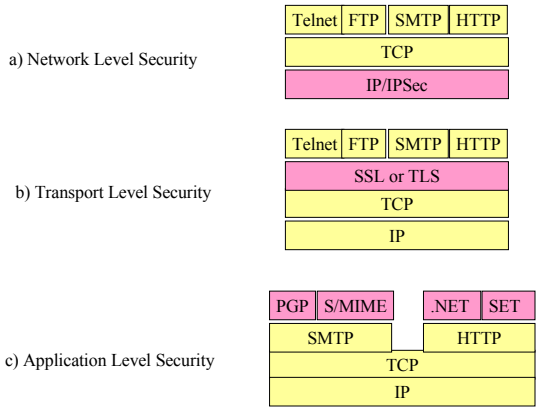


Figure 10-2; Approaches to Web Security (Based on Stallings [2000])

10.2.2 Secure Socket Layer (SSL) for Web Security

10.2.2.1 Overview

Secure Socket Layer (SSL), also known as Transport Layer Security (TLS), is by far the most heavily used security technology for the World Wide Web. At present, SSL is being packaged with almost all Web browsers (Netscape Navigator, Microsoft Internet Explorer) and servers (Apache, IIS). Historically, a competing protocol (S-HTTP) was also introduced roughly at the same time. In addition, the cryptographic principles of S-HTTP and SSL were the same (digital envelopes, signed certificates, message digests, etc.). However, S-HTTP is rarely used at present (see Section 10.2.3 for a discussion of S-HTTP).

SSL was introduced in a previous chapter (Chapter 6). The discussion here highlights how SSL is used to secure Web traffic and provides additional information. The key points about SSL are:

- SSL runs on top of TCP/IP and manages secure messaging on the network (see Figure 10-3). The SSL protocol provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection.
- The SSL protocol provides a wide range of encryption and authentication choices to ensure that communications between a client and a server remain private based on user requirements. The cryptographic choices are known as "cipher suites." A user can select a cipher suite when establishing an SSL session.
- To use SSL, you just need to type "https" instead of "http." For example, the link "https://www.fedex.com" connects you to the Federal Express Web site over SSL.
- Once an SSL session is established, all Web server-to-client traffic (both ways) is encrypted. Thus all SSL communications are encrypted including URLs of the requested document, contents of the requested document, contents of any filled-out forms, cookies sent from client to server, cookies sent from server to client, and contents of the HTTP header. Thus you can have secure Web communications.
- Figure 10-4 shows the exchange of messages between the two parties to establish an SSL session. The first few steps establish a secure connection. After step 9, an SSL session is established, and all Web server-to-client traffic (both ways) is encrypted.

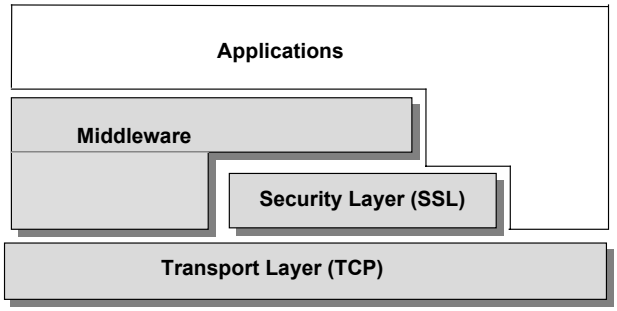


Figure 10-3: SSL

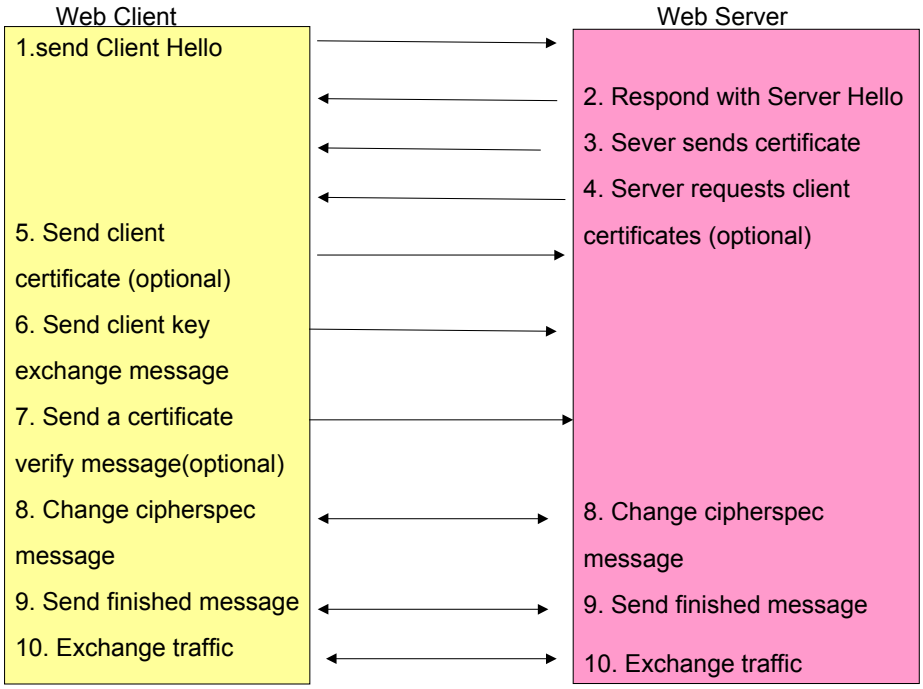


Figure 10-4: Flow of an SSL Session

10.2.2.2 A Closer Look at SSL

SSL uses public key encryption to provide security at the packet level. At the receiving end, SSL provides Server Authentication Message Integrity checks. As mentioned previously, the basic public key scheme assumes that both parties have a private/public key pair, and that they can trade these pairs. Then, they use their private keys to encrypt messages to be sent, and the public ones to decrypt received messages. Additionally, message integrity can be verified when checksum is generated for each packet, signed with the private key, and sent along with the packet.

However, this kind of encryption is too time-intensive. A faster encryption scheme consists of encrypting the main part of the packet, and then sending the key used to encrypt the packet. This key is itself encrypted using the private key and so is well-protected. This way, the heavy-duty encryption is used only for the "session key," which is very short, making the whole transaction close in speed to a non-encrypted one.

SSL adopts this latter scheme. A "master key" is generated using some random data. The master key is used to generate a session key for each session, from which a client write key and a server write key are generated (you read with the other party's write key). The server's public key is used to encrypt the master key during the initial handshake; from then on, the packets are encrypted with the server or client write key, depending on who is sending it; then a digest is taken, and the whole thing is finally packaged up with a session number. A faster version is obtained by using DES, a faster form of encryption than RSA.

SSL comes in two strengths, 40-bit and 128-bit, which refer to the length of the "session key" generated by every encrypted transaction. The longer the key, the more difficult it is to break the encryption code. Most browsers support 40-bit SSL sessions, and the latest browsers, including Netscape Communicator 4.0, enable users to encrypt transactions in 128-bit sessions – trillions of times stronger than 40-bit sessions. Global companies that require international transactions over the Web can use the global server certificates program to offer strong encryption to their customers.

In order to use SSL between an individual using a Web browser and a business hosting a website on a secured server, the following must occur:

- A Root Certificate must be installed on the individual's local Web browser. Certificate Authorities (CA) such as www.ssl.com provide Root Certificates for public downloading by individuals.
- A business must have a Server Certificate installed on their secured server. There are two ways a business can obtain access to a secured server: they can subscribe to a Secured Service Provider (SSP) such as ssl.com, or they can obtain a Server Certificate from a CA. In the latter case, the business must first ask their Internet Service Provider (ISP) to generate a Certificate Signing Request (CSR), and then, the CSR must be submitted to the SSP. The SSP will verify that the business is legitimate and will issue the business a Server Certificate. This certificate is then installed on the secured server, and SSL transactions are then enabled.
- Digital certificates encrypt data using Secure Sockets Layer (SSL) technology. Because SSL is built into all major browsers and Web servers, simply installing a digital certificate turns on their SSL capabilities.

For further reading, consult the SSL site (www.ssl.com). This site has comprehensive documentation on the mechanics of SSL and certificates.

10.2.3 S-HTTP (Secure HTTP)

S-HTTP (Secure HTTP) employs the same type of cryptographic techniques as SSL, but it is rarely used at present. Historically, SSL and S-HTTP came into existence roughly at the same time. The basic difference was that SSL was free (it was shipped free with the Netscape browser) but S-HTTP was not (it was part of NCSA Mosaic with a license fee). This distinction perhaps was the major deterrent to the popularity of S-HTTP. Another possible limitation of S-HTTP is that it is restricted to HTTP (it uses HTTP headers). Thus its use is limited to applications that are based on HTTP only. What to do with applications that do not run on top of HTTP, such as the applications that use CORBA or access remote data through ODBC/JDBC?

Although the use of S-HTTP so far has been rare, it may be brought back to life due to the increased popularity of XML-Web services. In particular, the Microsoft Dot Net initiative uses HTTP as the primary mechanism for remote procedure calls. For example, Dot Net is based on the XML-SOAP (Simple Object Activation Protocol) that uses HTTP as the transport mechanism.

10.2.4 Web Cookies – Some Privacy Issues

Web cookies, also known as cookies, raise several privacy issues. A cookie is a small data item that was introduced to maintain HTTP state[1] by Netscape around 1994. A cookie is created by a Web server and sent to a client (Web browser) as a header. It is used subsequently by the server. The client does not interpret cookies. Cookies are kept in browser memory, then written to a file for future communications. Figure 10-5 shows the basic operations of a cookie. The basic operations of cookies are:

- Client (Web browser) contacts a Web server for some information.
- Server responds but also installs a cookie at the client browser by issuing the command

  set-cookie:customer="joe," path="/cars."

- All future interactions with the client browser have the cookie.

  customer="joe," path="/cars"

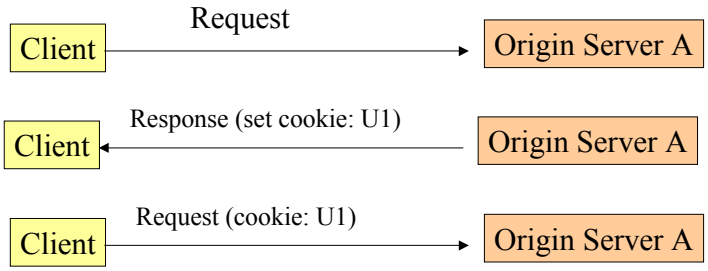


Figure 10-5: Cookies

Technically speaking, the Web server uses the cookies to maintain a session with the browser by noting the pages that the browser has visited. Naturally, this has raised several privacy issues because cookies can tell what Web pages have you visited, how long you visited them, etc.

Users, however, can control cookies to maintain their privacy. They can decide to have no cookies, accept cookies from specific sites only, or for a specific session only. There are some tradeoffs involved in these decisions. For example, if a client decides not to have a cookie, then the server may not be able to maintain a session with the client. A good approach is to clean up your cookies directory every now and then.

Cookies can be client-side or server-side, although most cookies are client-side. For example, the discussion of cookies so far has been on the client side. Server-side cookies can also be created by a Web server. In this case, a cookie is installed on the server and behaves as the "audit trail." Since the cookie is installed and runs on the server, the clients have no control over this cookie.

The Web site ww.cookiecentral.com has a great deal of information about cookies.

---

[1] HTTP, the protocol used between Web browsers and servers, is "stateless"", i.e., each interaction between a browser and a server is independent of the previous. This causes problems in building systems where you need to know what have you done previously.

10.2.5 Web Proxies for Security

Proxies sit between clients and servers and act as "standins." For example, a proxy acts as a server to a client and vice versa (see Figure 10-6). Proxies can be used for security, workload balance, etc. Proxy servers are commonly found in Web environments. A proxy server is essentially an intermediate program that behaves as a server but in fact passes the requests to real server(s). In effect, it is a fake server. In most practical cases, the proxy server receives the client calls, does some processing (typically security checking) and then, itself, becomes a client to other servers.

A proxy can handle security because it can sit outside a firewall and then pass information to a server inside the firewall. This does introduce security threats because a client not authorized for access may use a proxy to fake authorization. In addition to security, why else would anyone want to use proxies? A proxy can:

- Act on behalf of a server to provide anonymity to clients or servers
- Be a front-end to many clients – i.e., instead of handling calls from all clients, the Web server handles calls from the proxy that is front-ending many clients.
- Translate requests from Web to non-Web by intercepting calls from a remote program and converting them to Web HTTP calls (i.e., serve as a Web gateway)
- Cache the information so that the clients can access a local proxy instead of remote Web servers for information access
- Do workload balance by routing work to more than one server
- Filter requests and responses, if needed, between clients and servers

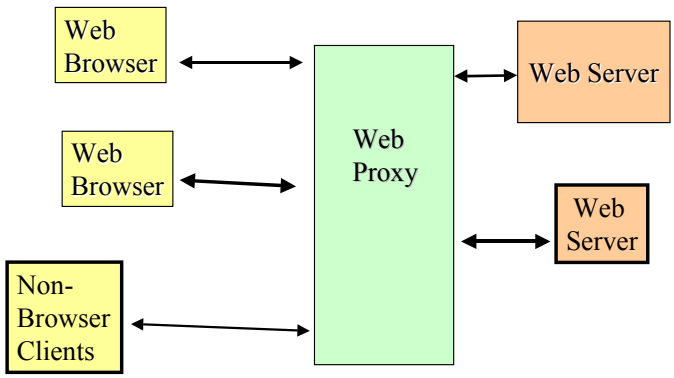


Figure 10-6: Web Proxies

Specific examples of using proxies are:

- A proxy server sits outside the firewall to receive the Public Internet calls, authenticates the clients, and then invokes services inside the firewall for authorized users. In many cases, these proxy servers rewrite packets to hide unneeded information.
- A proxy server, residing on an HTTP server machine, receives Java applet calls (recall that many browsers restrict Java applets to communicate with the HTTP server from where they were loaded) and then establishes connections with remote databases and programs on behalf of the Java applets.

- A proxy server behaves as an application gateway by receiving calls from different clients for different applications and then invoking needed applications.

Figure 10-7 shows how a proxy server is used between a customer and a seller. We are assuming here, for simplicity, that the customers reside entirely on the public Internet. The proxy server sits outside the seller firewall and connects to the seller purchasing system.

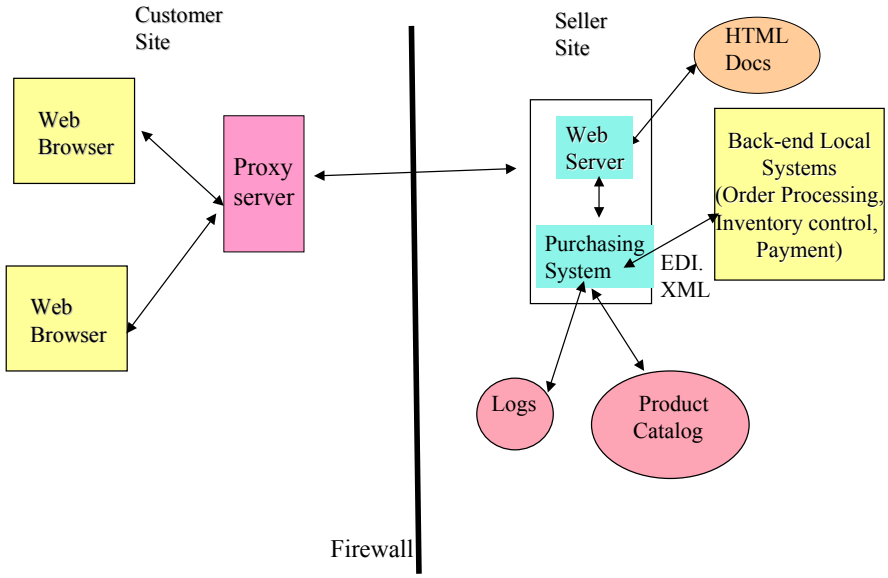


Figure 10-7: A Proxy Server Example

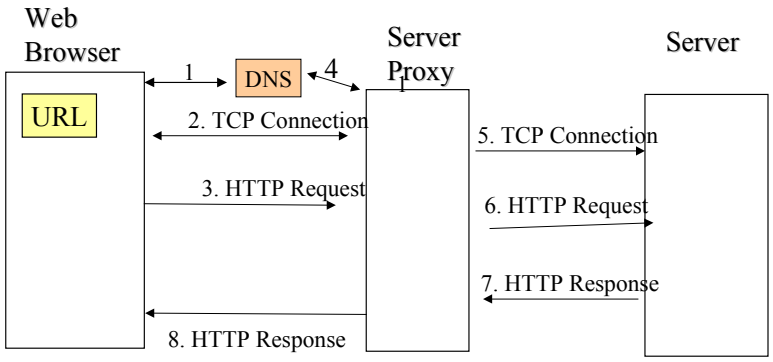
Figure 10-8 shows how proxy servers are used to locate Web servers. The browser uses the URL of the proxy server that responds and establishes a TCP connection to the proxy server through DNS and receives the HTTP request from the browser (steps 1, 2, 3). The main activity is in step 4 where the proxy server once again uses DNS to locate the target Web server. After the connection is made with the Web server (step 5), the proxy server sends the HTTP request received from the Web browser in step 3 to the Web server (step 6). The response is received and also sent back to the browser by the proxy (steps 7 and 8).

As expected, a wide range of proxy design issues need to be addressed:

- Security/privacy – the proxies should not promote/demote clients for their security status (i.e., the client security state must be faithfully carried forward by the proxy).
- Non-promotion/demotion – the proxies must stay at the server level and not assign themselves higher-level status.
- Handling large connections – proxies must be able to handle a large number of connections from clients.
- Providing audit trails – proxies must keep good audit trails to support non-repudiation.

- Handling cookies – proxies must know how to handle client- as well as server-side cookies.

For an extensive discussion of proxy design issues, see the book *Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement* by Balachander Krishnamurthy and Jennifer Rexford (Addison Wesley, 2001).



Step 4: The proxy uses DNS to locate the real server

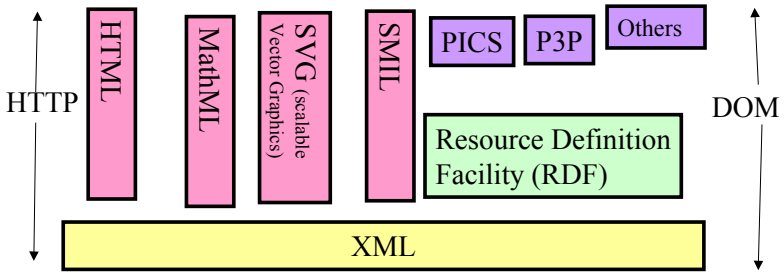
Figure 10-8: How Proxy Servers are Used to Locate Web Servers

10.3 Semantic Web and Semantic Web Security

10.3.1 Overview of Semantic Web

The first-generation Web, needless to say, gained popularity due to its simplicity and ease of use. However, the use of this technology has spread far beyond the initial design goals and imagination of the designers. Thus, some limitations of the initial Web technologies have started to show. The World Wide Web Consortium (known as W3C) has embarked on work in several directions to overcome these and other possible limitations. This work, under the general umbrella of Next Generation Web and now known as the Semantic Web, has introduced several new components to make the Web more powerful. However, Semantic Web also introduces new security issues that are not associated with the classical first-generation Web. The Semantic Web includes the following key developments (see Figure 10-9):

- XML (Extended Markup Language) family of standards and services introduced to improve machine-to-machine communications.
- Variations of markup languages such as HTML, MathML, SVGML, SMIL, etc. for different applications.
- Improvements in the core technologies such as improvement of HTTP and also introduction of more choices for Web gateways.
- Web automation efforts that introduce an object model for Web. These efforts include DOM, RDF, PICS, P3P and others.



PICS: Platform for Internet Content Specification
P3P: Platform for privacy preferences
SMIL: Synchronous Multimedia Interaction Language
DOM: Document Object Model

Figure 10-9: The Semantic Web Components

It can be seen from Figure 10-9 that XML is at the core of the Semantic Web. We will first review the motivation for the Semantic Web and then concentrate on various aspects of the Semantic Web security with emphasis on XML security.

What is the main motivation for the Semantic Web? At present, a great deal of information is being advertised by websites about the company products, special discounts, business partners, office locations, etc. Consider the situation where you wanted to use the Web to find out how many bookstores offer volume discounts, or what drugstores can fill certain prescriptions, or how many PC stores can repair your laptop in your neighborhood. Your main choice is to sit in front of your Web browser and issue endless queries, review the results manually and then make selections. Anyone who has used a Web search service like AltaVista or HotBot knows that typing in a few keywords and receiving a couple of thousand "hits" is not necessarily very useful. A lot of manual "weeding" of information has to happen after that.

It would be very nice to have an "automated agent" that could go around the Web, do all the work, and return the needed responses. These Web agents, while appealing, are not very easy to implement at present because the websites present information to you in a human-readable format. So far, the Web is mainly built as a forum for human interaction; because most Web documents are written for human consumption. The only available form of searching on the Web is to simply match words or sentences contained in documents. For example, by just looking at the HTML output, how can a program determine the location of a store? It will have to do "Web scraping," i.e., issue a query to a Web site, store the output received on a local page, and search the page for something that may sound like a location (e.g., city name). This is where XML and its variants have helped a lot. By using XML elements, the advertisers can define a tag "location" that can be used by agents to search and retrieve Web data. The benefits of structured Web data are not limited to intelligent personal agents. In fact, B2B trade can be greatly improved through automated programs that read, interpret, and react to websites of business partners.

Semantic Web, an initiative of the World Wide Web Consortium (W3C) led by Tim Berners-Lee in 1998, is addressing the major problem with the World Wide Web as it exists today – it is really hard to automate any tasks which one has to perform on the Web. A possible solution for the search problem – and for the general issue of letting automated "agents" roam the Web performing useful tasks – is to provide a mechanism that allows a more precise description of things on the Web. This, in turn, could elevate the status of the Web from machine-readable to something we might call machine-understandable, i.e., Semantic Web. Semantic comes from the Greek words for sign, signify, and significant, and relates to meaning. The Semantic Web, also known as the Next Generation Web, is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It is the idea of having data on the Web defined and linked in a way that it can be used for discovery, automation, and integration across various business and personal applications. The basic components of Semantic Web are:

- XML and its family members, to define, store, and exchange structured data
- DOM (Document Object Model), to read, process, and manipulate XML data through programs
- RDF (Resource Description Framework), to define the metadata (data about data) for the Web.

The following sections quickly review these components of the Semantic Web and discuss their security implications.

10.3.2 XML – A Quick Overview

10.3.2.1 What is XML and Why is it so Popular?

Nothing in the Semantic Web has gained as much popularity as XML. Widespread use of XML is also raising special issues. XML, as stated in Chapter 7, is a markup language for documents containing structured information. Here is an example of XML – it represents a computer located in Chicago:

```
<?xml version="1.0" standalone="yes"?>
< - computer example - >
<Computer>
<Location city="Chicago">
              <name>
                     <Type>Dell</type>
                     <ID>IPX-222599</ID>
              </name>
              <Properties>
                     <CPU>550 MHZ</CPU>
                     <RAM> 56 Meg</RAM>
                     <Disk>20 gigabyte </state>
                     <Modem>56 Kbps </zip>
              </properties>
              <price>$900</price>
       </Location>
</computer>
```

Although XML is relatively new, it has found applications ranging from e-commerce to government. Here are some examples of where XML is being used:

- E-commerce. XML is being used, of course, in e-commerce for representing the purchase orders, invoices, and inventories. This is an active area of work, with many standards such as ebXML, Rosettanet and Open Application Group at the forefront.
- XML-Web Services, also referred to as Web Services, are at the foundation of Microsoft .NET. These services are designed to build a very wide range of future Web-based applications that use the business component model. These components can be invoked over HTTP, thus making it a possible candidate for global Web applications. XML is at the core of these services as a mechanism for exchange of information between all players.
- Push technology and Web publishing. XML is being used to represent the programs (e.g., stocks, news, special offers, events) that the users can subscribe to. The foundation is an XML-based Channel Definition Format (CDF) that describes who subscribes to what (see the Web site www.iechannels.com for a sample).
- Online banking. XML is being used to represent financial transactions for online banking. An example is the Open Financial Exchange (OFX) specification. XML documents in OFX specify account information, monthly payments, bank statements, etc.
- Software distribution. XML is at the foundation of the OSD (Open Software Distribution) specification that allows users to download the software they need, with proper associated and prerequisite software.
- Web automation. XML is at the foundation of WIDL (Web Interface Definition Language) to generate programs that can read and react to the websites.
- Scientific publishing. XML is the foundation of several markup languages for scientific publishing. Examples are the Mathematical Markup Language and Chemical Markup Language.
- Airline reservation systems. XML is being used in the airline industry to represent flight information, arrival times, departure times, etc.
- Music and Theater. At present, several efforts are underway to use XML to represent some songs (title, singer, year of recording) and also some operas and plays.
- Government and defense. Many government applications are beginning to use XML at their core. Possible applications include the US Army (an Army markup language for exchanging information between various Army units), US Naval supply chains, and a US Virtual Spare Parts supply base.

Basically, many groups, outside and inside W3C, are defining new formats for information interchange. The number of XML applications is growing rapidly, and the growth appears likely to continue. There are many areas (for example, the healthcare industry, government, and finance) where XML applications are used to store and process data. Due to the high potential use of XML, it is important to analyze the security threats of XML.

10.3.2.2 The XML Family at a Glance

XML was initially defined in 1996 by the W3C. Since 1996, the work of several W3C working groups and other standards/industrial bodies has resulted in a "family" of XML standards that includes (see Figure 10-10):

- Document Type Declarations (DTD) to specify the set of rules for the structure of an XML document. DTDs are used to verify and validate XML documents and are thus central to XML-based e-business exchanges.

- XSL (eXtensible Stylesheet Language) to display information. XSL supports a basic premise of XML – separation of content from presentation. XSL is also useful for translating one XML description to another.
- XML query language to support querying of XML data
- XML schema to specify the format (e.g., the character lengths) and relationships between XML elements
- Other developments such as XML Link, XML Signature, and XML Path
- Variants of XML for different application areas. Examples are the Wireless Markup Language (WML), Voice Markup Language (VML), Mathematical Markup Language (MathML), Chemical Markup Language (CML) and others.

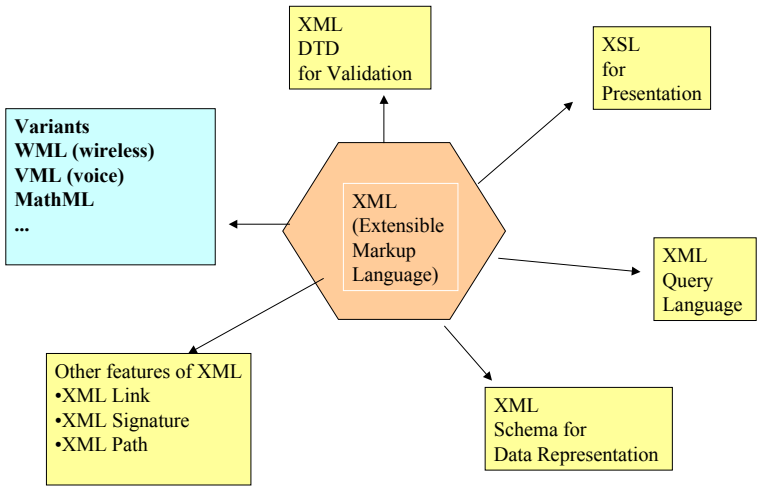


Figure 10-10: The XML Family

10.3.2.3 Document Type Declarations (DTD)

Document Type Declarations (DTD) are an important part of XML. A DTD specifies a set of rules for the structure of the document. For example, the following is a DTD for the customer record defined previously:

```
<!ELEMENT customer (name, address?, phone?)>
<!ATTLIST customer id CDATA #REQUIRED>
<!ELEMENT name (first, middle?, last)>
<!ELEMENT address (street+, city, state, zip)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT middle (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
```

In a DTD, optional fields are suffixed by a "?." For example "address?" and "phone?" indicate that address and phone number are optional, but the name is not. Thus, this DTD

will make sure that all XML documents that describe the customer have a name entry. Also, a multi-line entry is suffixed by a "+." (for example "street+" indicates that the street address may go over more than one line). The CDATA and PCDATA entries indicate that the element is character data or parsed character data, respectively. CDATA is just string data that is not parsed, while PCDATA is.

We can save this into a file called customer.dtd. DTDs are required in SGML but optional in XML. DTDs are the foundation of well-formed and valid documents:

- A well-formed document adheres to the syntactic rules defined by the XML standard – e.g., tags are delimited by < and >.
- A valid document is a well-formed document that also adheres to the rules of a specified Document Type Definition (DTD).

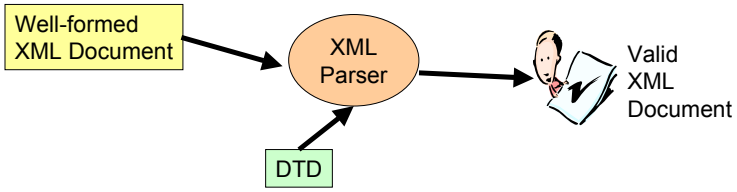


Figure 10-11: The Role of XML

Thus, if you create an XML document without a DTD, then it is well-formed but not valid. By creating a DTD, you make the document well-formed as well as valid because XML parsers compare the document against a DTD to verify if the document adhered to the DTD specifications. In e-commerce, it is always a good idea to define a DTD before creating XML documents.

The following XML definition represents a valid XML document (note that we have specified that this document is not standalone and that customer.dtd is consulted as a DTD):

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE customer SYSTEM "customer.dtd">
<customer id="12345">
        <name>
                <first>Amjad</first>
                <last>Umar</last>
        </name>
        <address>
                <street>MCC-1C337B</street>
                <street>445 South Street</street>
                <city>Morristown</city><state>NJ</state>
                <zip>07960</zip>
        </address>
        <phone>973-829-3114</phone>
</customer>
```

The following document represents a well-formed but not valid XML document:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE customer SYSTEM "customer.dtd">
<foo1 id="12345">
        <foo2>
                <foo3>Amjad</foo3>
                <foo4>Umar</foo4>
        </foo2>
```

```
            <foo5>
                  <foo6>MCC-1C337B</foo6>
                  <foo7>445 South Street</foo7>
                  <foo8>Morristown</foo8><foo9>NJ</foo9>
                  <foo10>07960</foo10>
            </foo5>
            <foo11>973-829-3114</foo1>
</foo1>
```

10.3.3 XML Security, XML Encryption and XML Signature

10.3.3.1 Issues in XML Security

XML has introduced several security concerns. The main concern is that intrusion of DTDs or Schema can have a very serious impact. For example, trading on hubs that use XML can completely stop if the XML DTDs/schemas are modified to invalidate the transactions. It is important that the important XML documents and DTDs be properly protected through encryption and authentication/authorization controls.

A straightforward way of securing an XML document and DTD is to encrypt it and then use some type of message digests for integrity before transmitting over the network. This can be, and is done, on a regular basis by using SSL or TLS. However, different parts of the same XML document need different levels of security because of the following situations:

- Complete XML documents can be sent once and then accessed by multiple people with different needs on the other side.
- Different groups of users may need different parts of the XML document. For example, a merchant needs to know a buyer's name, the items bought and shipping address but does not need the credit card information. However, the bank needs to know the credit card information but has no interest in the goods bought.
- Different parts of the same XML document may need different digital signatures. For example, a contract between three parties may require signatures in three different sections of the same XML document.
- If an XML document is encrypted as a whole, including the tags, then the DTD or schema cannot be used to parse or search the data relevant to the encrypted tags.
- If the XML tags of a document with well-known DTD/schema are encrypted, then these encrypted tags may be "reverse engineered" by hackers to determine the cryptography employed. For example, if a purchase order based on a standard (e.g., Rosettanet) is encrypted along with tags, then an encryption key can be quickly determined by a hacker because the Rosettanet DTD for a purchase order is well known and publicized.
- Integrity checking of XML is also non-trivial. Any change to an XML document to which a cryptographic hash algorithm has been applied raises integrity alarms. However, two XML documents may differ in exact textual comparison though they may be logically equivalent. This is because comments, line delimiters, empty tags, and other ways of representing information can vary between two XML documents with the same logical information. For example, inserting a comment would indicate a modification even though it does not change the meaning of a document.

Thus XML security needs to support different types of encryptions, message digesting and digital signatures. In essence, every XML element may need a different cipher and/or a different digital signature. The following discussion concentrates on protecting sensitive information in the XML documents, establishing integrity, and authenticating the source of different parts of such documents. There are other closely associated areas, such as authenticating users or systems, identifying levels of authorization, and managing keys, that are also relevant to XML security. In particular, Web Services security and SAML (Security Assertion Markup Language) are of vital importance. We will discuss SAML in the next chapter. eXtensible Access Control Markup Language (XACML or XACL) has been defined for access privileges of a particular XML document. In addition, work is continuing on establishing a protocol for key management on top of the XML signature standard. XML security is still evolving but progress is being made in several areas. This discussion of XML security is short and intended to highlight the key developments. For additional information and ongoing developments, visit the W3C website (www.w3.org).

10.3.3.2 XML Encryption for Privacy

To illustrate XML encryption, let us consider an XML file that represents patient information (name, age, medical condition, hospital admission information, and health insurer). This information needs to be viewed by different people with different authorities and needs. In addition, the privacy rights of the patients need to be protected. For example, a doctor or nurse may need medical details but not all personal materials such as health insurer; a medical researcher should be prevented from seeing any personal details such as patient name; and a hospital administrator may need to see admission and insurer information but should be prevented from viewing medical history. Let us use the following simplified XML document as a sample XML. In this listing, xmlns statements specify "XML Namespaces" for name conflict resolution. A namespace is a way of identifying a part of the Web space from which we can derive names. By using namespaces, anyone can create their own tags and mix them with tags made by others.

```
<?xml version='1.0'?>
<PatientInfo xmlns='http://example.org/patientv2'>
<Name>Joe Donovan<Name/>
<Age> 52 </Age>
<Medical-History>
<Medical-Condition> High Blood Pressure </Medical-Condition>
<Medical-Condition> One Heart Attack </Medical-Condition>
</Medical-History>
<Hospital-Admission>10-5-2003, Omaha Hospital</ Hospital-Admission >
<Insurer> CIGNA</ Insurer >
</PatientInfo >
```

To encrypt this information, XML encryption provides many options. The basic element is the EncryptedData element that can be used to encrypt any XML element, an XML element content, an entire XML document or any arbitrary data in an XML document. The encrypted data is an XML encryption element that contains or references the cipher data. In the following example, we have encrypted everything other than the patient

name and his insurer. The results are stored in a CipherData element. Note that all tags as well as data are kept private by encrypting them.

```
<?xml version='1.0'?>
<PatientInfo xmlns='http://example.org/patientv2'>
<Name>Joe Donovan<Name/>
<EncryptedData Type='http://www.health.org/xmlenc#Element'
     xmlns='http:// www.health.org /xmlenc#'>
  <CipherData><CipherValue>523A7DD22U</CipherValue></CipherData>
</EncryptedData>
<Insurer> CIGNA</ Insurer >
</PatientInfo>
```

In the following case, we have only concealed the medical history and hospital admissions from unauthorized people. Note that the tag names (<Medical-History> and <Hospital-Admission>) relevant to the encrypted content are shown but the actual information is encrypted. This could be useful to verify that the document is complete while some information is hidden.

```
<?xml version='1.0'?>
<PatientInfo xmlns='http://example.org/patientv2'>
<Name>Joe Donovan<Name/>
<Age> 52 </Age>
<Medical-History>
<EncryptedData Type='http://www.health.org/xmlenc#Element'
     xmlns='http:// www.health.org /xmlenc#'>
  <CipherData><CipherValue>252C39ABC3</CipherValue></CipherData>
</EncryptedData>
</Medical-History>
<Hospital-Admission>
 <EncryptedData Type='http://www.health.org/xmlenc#Element'
     xmlns='http:// www.health.org /xmlenc#'>
  <CipherData><CipherValue>AXZ3225XU</CipherValue></CipherData>
</EncryptedData>
</ Hospital-Admission >
<Insurer> CIGNA</ Insurer >
</PatientInfo >
```

In some cases, it may be desirable to encrypt all information in the document. The following example shows this.

```
<EncryptedData Type='http://www.health.org/xmlenc#Element'
     xmlns='http:// www.health.org /xmlenc#'>
  <CipherData><CipherReference>"http://www.health.org/p1"></CipherRe
ference></CipherData>
</EncryptedData>
```

Note that in this case, a CipherReference element is used to represent encrypted data. In this case, a URI points to the location of the encrypted data. In the previous cases, the encrypted data was shown by the contents of the CipherValue element. Thus a CipherData element can either envelop or reference the raw encrypted data. The referenced data approach can be more secure because the encrypted data can be kept at another secure site with proper authentication controls instead of being enclosed as part of the XML document.

10.3.3.3 Canonical XML for Integrity Controls

XML encryption helps with the privacy of XML but does not guarantee integrity – an intruder could still modify the encrypted data to make you miserable. Mechanisms are needed to determine if parts of XML documents have been modified. In particular, it is important to know if the meaning of the document has been changed. As indicated previously, two XML documents may differ in exact comparison though they may carry the same information because of the differences in comments, line delimiters, and empty tags. The canonical XML specification was developed to address this problem. This specification describes a physical representation of a document, known as the canonical form, that allows two documents with the same canonical form to be considered logically equivalent, allowing for textual differences.

The main idea is to generate message digests from the canonical forms. For example, different tools (such as parsers) could generate different texts and thus different message digests. However, if message digests were created from canonical forms instead of the texts, then the same message digests would be created. This is particularly important for digital signatures because textual variants in signatures should not violate the integrity of a document or the authentication of its sender.

10.3.3.4 XML Signatures and the Web of Trust

In XML-based e-commerce, parts of a document may need to be signed, perhaps by different people. A signer should also be able to view the item to be signed in plain text – this may mean decrypting part of something already encrypted for other reasons. XML signatures can be applied to any arbitrary data content, encrypted or not. The signatures in which the data is external to the signature element are termed detached signatures. On the other hand, the signatures applied to data within the same XML document as the signature are known as enveloping or enveloped signatures. The following is an overly simplified example of a detached signature.

```
<Signature Id="MySignature"
        xmlns="http://www.ngesolutions.com/xmldsig#">
 <SignedInfo>
  <Document> Invoice </Document>
</SignedInfo>
<SignatureValue>TTF039U3O</SignatureValue>
<KeyInfo>
   <KeyValue> P2P37FU </KeyValue>
</KeyInfo>
</Signature>
```

The data that is actually signed is in the SignedInfo element (the invoice). In reality, this document may be quite long. The signature element is outside the signed section and is represented by the SignatureValue and KeyInfo elements.

The issue of trust is closely related to digital signatures. Even if the signature is authenticated, you may not trust the signer. For example, if Bob gives a signed "IOU" for $10,000, you may still decide that you don't trust Bob for such an amount (well, Bob has not paid his previous debts!). Just as you can refuse to trust Bob's signature, you can also tell your program whose signatures to trust and whose not to. We can set our own levels of trust (or paranoia) to assure that the programs can decide what to believe. Trust levels can be specified by using RDF (see Section 10.3.7).

"Web of Trust" is used to specify how trust can be transferred on the Web. The notion is similar to "a friend of you is a friend of mine." For example, you tell your program that you trust your best friend, Sam. However, Sam trusts quite Pat and a number of other people. And of course, Pat trusts another set of people. Each of those people trust other people, and so on. The trust relationships fan out quickly, forming a "Web of Trust." Each relationship has a degree of trust (or distrust) associated with it.

A program takes these factors into account when deciding how trustworthy a piece of information is. The process can be transparent or opaque. For example, you might be happy with a simple "thumbs up/thumbs down" display or may insist on seeing some or all of the trust factors involved in the decision. Tim Berners-Lee has proposed an "Oh, yeah?" button, that when clicked would provide reasons why you should trust the data. The information necessary to make a trust decision is based on the Web of Trust specification.

10.3.4 The "XML-ing" of PKI

PKI, as discussed in a previous chapter, is a family of technologies that are based on the public key cryptography; it contains the facilities to store and manage certificates (i.e., the ability to issue, revoke, store, retrieve, and trust certificates). Specifically, PKI capabilities help create and manage asymmetric cryptographic keys or public/private key pairs required by applications. Since PKI requires storage, retrieval, and validation of documents (digital certificates, for example), XML can be used to describe PKI documents and operations on the documents. XML Encryption and XML Signatures are good steps in this direction because they can be used to encrypt and sign documents. Thus XML can be used to apply security and PKI functions to portions of documents such as digital certificates. In addition, vendors are trying to convert certificate-handling routines to XML-based applications to perform functions such as requests, certificate submittals and validations based on various protocols. There are potential benefits of combining XML with PKI, but too many standards can cause confusion. Stay tuned.

10.3.5 Web Metadata and PICS (Platform for Internet Content Selection)

Metadata is "data about data" or specifically in our current context, "data describing Web resources." For example, the site map of a Web site shows what Web pages are on the

Web site and how they are interlinked. Thus, a site map is metadata of a Web site. The distinction between "data" and "metadata" is not an absolute one; it is a distinction created primarily by a particular application ("one application's metadata is another application's data").

The history of Web metadata is long at W3C, beginning with PICS (Platform for Internet Content Selection). PICS is a mechanism for specifying ratings of Web pages through rating labels. These labels are XML tags that contain information about the content of Web pages: for example, whether a particular page contains sex, nudity, violence, foul language etc. These tags can be used to determine, for example, if the page is suitable for viewers under some age. Recall that XML is being used to produce the next generation of *TV Guide*. Thus, the PICS labels could be used to determine ratings for TV viewers. Similarly, parents worried about their children's Web usage could set their browser to filter out any Web pages not matching their own criteria. PICS development was motivated by the possible restrictions on the Internet through US legislation such as the Communications Decency Act.

PICS is founded on the belief that individuals, groups and businesses should have easy access to content selection products based on a diversity of voluntary rating systems. The PICS Working Group is trying to facilitate the following:

- **Self-rating**: enabling content providers to voluntarily label the content they create and distribute.
- **Third-party rating**: enabling multiple, independent labeling services to associate additional labels with content created and distributed by others. Services may devise their own labeling systems, and the same content may receive different labels from different services.
- **Ease-of-use**: enabling parents and teachers to use ratings and labels from a diversity of sources to control the information that children under their supervision receive.

Members of the PICS Working Group include businesses from the computer, communications, and content industries. In addition, trade associations and public interest groups are part of the PICS effort. For more information about PICS, see (www.w3.org/PICS/).

PICS is a limited metadata framework. RDF, described later, is a general metadata framework – and in a way is a general knowledge representation mechanism for the Web.

10.3.6 Platform for Privacy Preferences (P3P)

P3P was developed by the W3C for the consumers to gain more control over the use of personal information on websites they visit. P3P is basically a set of multiple-choice questions that cover the major aspects of a Web site's privacy policies. These policies show how a site handles personal information about its users. For example, a Web site may have the policy that it will display the names of the people who visit the site but not what they accessed. This policy is represented in P3P and stored in a P3P-enabled Web site. This site makes this information available in a machine-readable format that can be read by P3P-enabled browsers. These browsers can interpret and compare these preferences to the consumer's own set of privacy preferences. Thus P3P allows privacy

policies to be explicitly specified and stored where they can be found. This allows the users to base their actions on the specified policies.

What do the P3P statements look like? Here is an example of a P3P statement posted by the CoolCatalog (www.TheCoolCatalog.com):

> The CoolCatalog of 123 Main Street, Bethesda, MD 20814, USA, makes the following statement for the Web page at http://www.TheCoolCatalog.com/catalog/. We have a privacy seal from PrivacySeal.org. Our privacy policy is posted at http://www.TheCoolCatalog.com/PrivacyPractice.html. We do not provide access capabilities to information we have about you.
>
> We use cookies and collect your gender, information about your clothing preferences, and (optionally) your home address to customize our entry catalog pages and for our own research and product development. We retain this information indefinitely.
>
> We also maintain server logs that include information about visits to the http://www.TheCoolCatalog.com/catalog/ page, and the types of browsers our visitors use. We use this information in order to maintain and improve our Web site. We retain this information indefinitely.

A W3C page (www.w3.org/P3P/compliant_sites) lists a large number of sites that are explicitly displaying their P3P policies. Here are a few examples:

- Innovative Nurses & Sitters (http://www.innovativenursing.com/)
- Ontario Equity Mortgages (http://www.ontarioequity.com/)
- iFriends (http://ifriends.net/)
- Diet Pills Users Forum (http://www.bontril-meridia-phentermine-xenical.org/)

A general example of P3P-type policies is the Microsoft .NET Passport (discussed in the next chapter). Businesses supporting Passport on their websites sign the nonexclusive .NET Service Agreement. Part of this contract states that the participating sites must post a privacy statement online and make it readily accessible to their users. Privacy policies are being posted by different companies. Privacy policies for Microsoft.com can be found at http://www.microsoft.com/info/privacy.htm and for .NET Passport at http://www.passport.net/consumer/privacypolicy.asp.

10.3.7 Resource Definition Framework (RDF) and RDF Security

The Resource Description Framework is a foundation for processing metadata (data about data). Let us use the example of site maps. Companies have a great deal of information on their websites. A company website typically contains advertisements, white papers, announcements, job openings, product descriptions, etc. Can you describe what is on your website in a systematic fashion so that an agent can understand it? RDF provides the facilities to describe Web sitemaps, privacy practices, digital signatures, device characteristics, etc.

RDF uses XML as its encoding syntax. The resources being described by RDF are, in general, anything that can be named via a URI (Uniform Resource Identifier). URIs are similar to URLs. You can describe the resources in terms of properties such as "author," "publisher," etc. For example, you can describe a book in terms of its title, publisher, author, chapter titles, etc. Some standard vocabularies are available for describing typical resources. The Dublin code (DC) is one of the earliest vocabularies that consists of:

Title
date

subject

Warwick code is another vocabulary.

Here is an example of how an announcement document about new benefits for personnel can be defined by using RDF:

```
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 Xmlns:DC="http://purl.org/dc/elements/1.0/"
<Description about = "http://www.telcordia.com/announcel.html"
<DC:title> Telcordia Personnel </DC:title>
<DC:date> 2000-10-10 </DC:date>
<DC:subject> New benefits </DC:subject>
</description>
</RDF>
```

As explained previously, XML Namespaces (specified in the xmlns statements) are used for name conflict resolution in RDF. The Dublin code (DC) is used in this example. The statement, "Xmlns:DC=http://purl.org/dc/elements/1.0/" shows that the Dublin Code vocabulary is used in this example.

The overall purpose of RDF is to provide meaning to the statements that are found on the Web. For example, the statement "Umar likes friends" could mean that Umar likes to have a bunch of friends, or that Umar likes the TV series "Friends." For a program to understand the difference, some of these terms need to be clarified. This is typically done through namespaces.

In general, a RDF statement is a very similar to a simple sentence, except that almost all the words are URLs. This makes the statements harder to understand when you first see them. In addition, each RDF statement has three parts: a subject, a predicate and an object. For example, the RDF statement says "Umar likes friends":

```
<http://www.amjadumar.com/>
<http://love.example.org/terms/Likes>
<http://www.TVGuide.com/Programs/Friends>
```

The statement above is written in N-tuples, a language that allows you to write simple RDF statements. The URL defines Umar, the second defines "likes" and the third "frineds" (the TV programs). If you think this statement is difficult, the official XML representation of RDF is more complicated, but says the same thing.

Naturally, it is difficult to imagine that human beings will be writing these statements. In general, software tools will create RDF statements. An example of such a tool is the RDF Gateway from Intellidimension (http://www.intellidimension.com/). RDF Gateway allows applications to query and store information as RDF data and also allows automated extraction of data from a wide range of external data sources.

Several security issues are related to RDF. The main problem is that the namespaces should be protected; otherwise the meaning of an RDF document can be modified completely. For example, a namespace may specify a vocabulary www.mything.html. However, if someone changes the content of this vocabulary, then the entire document is

suspect. Another issue concerns how a piece of RDF can be validated, analogous to the way that an XML document can be validated with a DTD. Since there is no equivalent of a DTD or schema for RDF, the answer is not straightforward.

For evolving issues and approaches to RDF capabilities and security, consult the W3C site (www.w3.org/RDF). A book (S. Powers, *Practical RDF*, O'Reilly & Associates, July, 2003) also has a great deal of detailed information about RDF.

10.3.8 Document Object Model (DOM) and DOM Security

W3C's Document Object Model (DOM) is a standard internal representation of a document structure and aims to make it easy for programmers to access components and delete, add or edit their content, attributes and style. The goal of DOM is to allow programmers to write applications that work properly on all browsers and servers, and on all platforms. While programmers may need to use different programming languages, they do not need to change their programming model. DOM allows programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page.

Let us go through an example to illustrate the key points. Figure 10-12 shows an "object model" of the following XML document that represents a customer.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE customer SYSTEM "customer.dtd">
<customer id="12345">
       <name>
              <first>Joe</first>
              <last>Zombie</last>
       </name>
       <address>
              <street>Graveyard Rd</street>
              <city>Horrorville</city><state>NJ</state>
              <zip>99999</zip>
       </address>
       <phone>555-888-1311</phone>
</customer>
```

If you want to write a program that processes this document, then the program will have to be written for this specific model. The basic idea of DOM is to present a generic model of XML and HTML that can be used by programmers to read and process documents.

DOM attaches behavior to Web documents – the Web documents are viewed as objects with a collection of nodes. Each node scan has attributes and text associated with it. In addition, nodes can be interlinked with a parent-child relationship. For example, each node can represent an XML element that is connected to sub-elements through parent-child relationships. Figure 10-12 shows the DOM view of the customer's XML document. A program can now travel through this tree to access and process various elements.

DOM provides an OO API for accessing Web documents and treats document contents as programmable objects. The class hierarchy is accessed by the API that defines standard traversing verbs such as next, child, parent, etc. DOM works in the following manner:

- Processor (browser) reads XML/HTML into an internal format and constructs the DOM tree.
- Client code (typically Java applet) reads the document contents.
- The client can store the document (entirely or in part) in a database, display it in different formats, or do anything else.

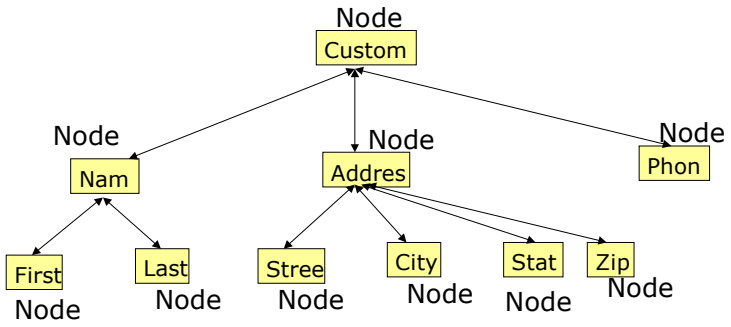


Figure 10-12: Object Model of Customer XML document

The Document Object Model (DOM) has the following potential security risks:

- It supports third-party extensions. If these extensions do not use proper security and authentication procedures, they could introduce viruses and bugs.
- It allows external entity references. If proper security and authentication procedures are not used, these references could also introduce problems.
- It is designed to run over the public Internet. If the security of DOM is compromised, it could expose several documents being processed by DOM to intruders.

The best approach to deal with DOM security problems is to disable external references and disallow external entity references. Disabling external references prevent the XML parser from retrieving information not contained in the XML document itself. In addition, it is a good idea to use a security manager such as the Internet Explorer Security Manager when invoking DOM from a browser.

10.4 Short Case Studies and Examples

10.4.1 Web Security at Exostar

Members of the defense business are obviously concerned about security breaches. In August 2002, a 23-year old hacker nicknamed RaFa broke into a highly secure NASA database, and stole 43 megabytes of sensitive design data about planned space vehicles.

The documents were created during a collaborative effort by Boeing, Aerojet and Pratt & Whitney.

So when companies such as Boeing, Rolls Royce, Lockheed Martin, BAE Systems and Raytheon invested $100 million to create the online aerospace marketplace Exostar in 2000, they chose to implement 87 security requirements. This included data encryption not only while data is in transit but also while it is stored in the database. Exostar also maintains detailed background information on each user and a 12-month record of every file being accessed, what changes were made and by whom; it allows no one person or company (even at Exostar) to have complete access to all the data. Many vendors collaboratively built the Exostar security system.

Rolls Royce put Exostar to the test by using Exostar's electronic collaboration services. By using these services, its engineers could securely share CAD patterns and project management systems with other design engineers at Fiat-Avio, Goodrich Corporation, Honeywell, Volvo, and others. At the start of the project, Rolls Royce appointed a project manager to initiate the process. The manager logged onto the system to start a session. Then Verisign verified the project manager's identity and authority to work on the project. Verisign gave the project manager a password and a digital certificate. The certificate resided on the manager's computer so it was only possible to access the system from that computer. The manager then invited others to join the project and specified the level of access to which each user was entitled. After initial processing, the partners from different sites were ready to share information and worked on the same document using their personal digital certificates for verification, while the file itself was encrypted with a 128-bit key. After the session ended, the file was then sent to Exostar's data center, for high levels of physical and network security. Whenever another project member wanted to access the file for revisions, it was encrypted again before traveling over the Internet to his computer, where it remained encrypted until the engineer with the authorized key opened it.

This security solution is not an overkill if you are in the defense business. Exostar has over 11,000 members who think that the extra security is worth it. It is estimated that by collaborating over the Web, Rolls Royce saved as much as 60% on its travel budget, reduced project management errors by up to 50%, and cut the product development cycle time by up to 40%. In addition, so far, no break-ins have been reported.

Source: Niall McKay, "Case Studies: Digital Do-Overs – Web Security: Xtreme Exostar," *Forbes*, 7 October, 2002

10.4.2 Jitux.A Worm Targets MSN Messenger Users

Jitux.A is a new worm in the form of an instant message (IM) inviting users to click on a URL which downloads the jituxramon.exe file. The file then becomes resident in the computer's memory and sends new messages containing the URL link every five minutes to all contacts stored in MSN Messenger. The virus, detected in Spain, Mexico and Portugal, seems to have no other apparent destructive effects, nor does it cause changes to the system configuration. The malicious code is wriiten in Visual Basic and runs on Windows 95, 98, ME, NT, 2000 and XP operating systems. Jitux.A is an addition to the more than 60 IM vulnerabilities that have been published by the security researchers from Symantec.

Source: J. Blau, "Jitux.A Worm Targets MSN Messenger Users," Accessed from Computerweekly.com, Tuesday 6 January 2004.

10.4.3 A Global Financial Services Company Uses XML Signature

ORIX, founded in Japan in 1964, is a global financial services institution that maintains a wide range of operations in both corporate and retail markets. The company offers leasing, lending, rentals, life insurance, real estate financing and development, venture capital, and many other financial services.

ORIX viewed the Internet and XML as a way of conducting B2B trade but was concerned about security. ORIX was especially concerned about non-repudiation. Digital signatures fulfilled this need because they can be used for authenticating the sender of a message, preserve integrity of the data, and support non-repudiation. In addition, digital signatures are legally binding in Japan because that country enacted the acceptance of digital signatures in April, 2000.

To respond to customer demands, ORIX moved quickly to implement digital signatures and accepted the IBM XML Security Suite that provides capabilities such as digital signature, element-wise encryption, and access control. By using this feature, ORIX customers could use the digital signature functions to digitally sign an XML document. However, the digitally signed XML document, when received by ORIX, had to be in a format that could be understood by ORIX applications. IBM designed a signature service, a proxy, that would mask the use of digital signatures from ORIX's applications. The proxy intercepts digitally signed messages and routes them to the signature service. If the signature is valid, the signature Web service removes the digital signature and associated information and then sends it to the back-end applications. After processing, the outbound message is routed to the signature server where it is digitally signed before being sent to the customer.

Source: http://www-306.ibm.com/software/ebusiness/jstart/casestudies/orix.shtml

Source of Information for Web and Semantic Web Security

- Oppliger, R. *Security Technologies for the World Wide Web.* Artech, 2000.
- Stallings, W. *Network Security Essentials*. Prentice Hall, 2000.
- Stein, L. *Web Security: A Step-by-Step Reference Guide*. Addison Wesley, 1998.
- Garfinkel, S., and Spafford, G. *Web Security & Commerce*. O'Reilly, June 1997.
- http://www.w3.org/Security/Faq/ – a website for The World Wide Web Security FAQ (maintained by W3C).
- SSL site: www.ssl.com

10.5 Suggested Review Questions

1) What are the main issues in Web security and what are the solution techniques?

2) What is SSL and how is it related to S-HTTP? Which one is used more often?

3) What are the issues unique to XML security?

4) What are the main approaches and standards being used to address the XML security issues?

5) Define different XML encryption schemes for an XML document that contains the following information (name, amount of purchase, items purchased, credit card number, credit limit, expiration date).

6) Surf the Web to find some interesting PICS and P3P specifications.

7) Look at the W3C site and identify the security developments that have not been discussed in this chapter.