Design and Development of Protective Module for the Transmission of Data Using a PLUG-IN Program

Seung Ju Jang,

Dongeui Univ. Dept. of Computer Engineering, Korea

Summary
This paper provides a client environment in which web pages can be used safely through document securing function among Internet web browser (Netscape Communicator or Netscape Navigator) functions. Transmission of cryptographic web data is performed by using the protective module. One characteristic of the present protective module is that it has an advantage of providing the web cryptographic function in an exclusive environment regardless whether the security environment is used in terms of software or in terms of hardware. Generally, a serious requiring of security environment is the case in which Intranet is implemented. In such Intranet environment, it is possible to provide an exclusive cryptographic function if the protective function proposed in the present paper is used. It is also advantageous in that the protective function can be used conveniently even in a general Internet environment. In order to satisfy security in the Internet environment, both the server and client should have protective modules in the present paper.

Key words:
plug-in, transmission,

1. Preface

A plug-in program function is a very convenient function for users of Internet. It is possible for users to program easily a function to be added without the source code of a specific program as the plug-in program function appears.

A plug-in program function is a function provided by the Netscape navigator. A Netscape navigate plug-in is the code portion of the browser as well as the code module which is loaded dynamically. That is, if a plug-in is loaded, it becomes a direct portion of the browser code. While this technique can provide the highest possible speed, it lacks security and independency of the platform. As a plug-in is like a web page formed without any fault, users sometimes do not know whether the plug-in is being operated after a proper plug-in is brought in.

Since a plug-in module is an object code, it should be written in a language which can compile them completely. Particularly, in case of Window, the development environment should be able to produce DLL (Dynamic Link Library) and be able to be completed with a compiled resource.

It is recommended to use Visual C++ language, if possible, for a language which can be formed by using the plug-in module. It is because the plug-in module is initially developed in the Visual C++ environment. Of course, it is possible to write in the Java language [6, 7, 8, 9].

A plug-in function is useful when desiring to develop application programs such as image viewer, document viewer, presentation, animation, 3-D, audio, video, etc. in the Netscape navigate web browser.

Described in the present paper are design and development of a web protective module using the plug-in function of the Netscape navigator. Recent rapid spread of Internet has seriously necessitated a web environment which can guarantee safe transmission of data for users. Provided in this paper, therefore, is a protective module using the Netscape navigator web browser in order to offer such environment readily by using the plug-in program function.

Existing commercial web browsers (Netscape Communicator, MS-Explorer, etc.) employ methods of solving problems from the point of view of software using SSL (Secure Socket Layer) protocol in offering cryptographic functions. However, such solutions from the point of view of software for materials sent and received in the Internet web environment make it difficult to approach and control problems strongly. Conventional protocols of solving web security using the software method include SSL, SEA, S-HTTP, etc. The most widely used software security protocol at present is SSL.

2. PLUG-IN Program Structure

The format used in the Netscape navigator for sending and receiving data between the server and client is of MIME type. The MIME (Multi-purpose Internet Mail Extension) type is manufactured in order to deal with additional data types in electronic mail and is used as an international standard. The plug-in is registered by the navigator using MIME information in which the resource is built. MIME defines the following header fields:

Manuscript received September 5, 2008.
Manuscript revised September 20, 2008.

MIME Version :
Content Type :
Content - Transmission – Encoding :
Content – ID :
Content – Technology :

The plug-in structure of the Netscape navigator is operated based on code modules which are loaded dynamically. These modules should be in the plug-in directory or sub-folder which is read by the navigator while the navigator is initialized. If the navigator finds the MIME type which is built in the web through HTML, it is loaded on a proper code module [1].

In case of an embedded plug-in, the HTML EMBED tag informs the navigator of the size of the plug-in window on the web page. This window produces a navigator instead of the plug-in. Instance of the plug-in is produced by calling NPP_New API while the plug-in is loaded. Generally, one web page has instance of one plug-in many times since the plug-in may be loaded many times while producing multiple instances. Data stream is a very important concept in the navigator plug-in structure. Most plug-ins receive data from the server in order to process, but in some other cases, request data in the random processing form. The navigator also provides LiveConnect. LiveConnect expands the plug-in structure while providing communication among the plug-in, Java Applelet, and Java Script. Java Runtime Interface (JRI) assumes an important role among them. The navigator plug-in structure is shown below:

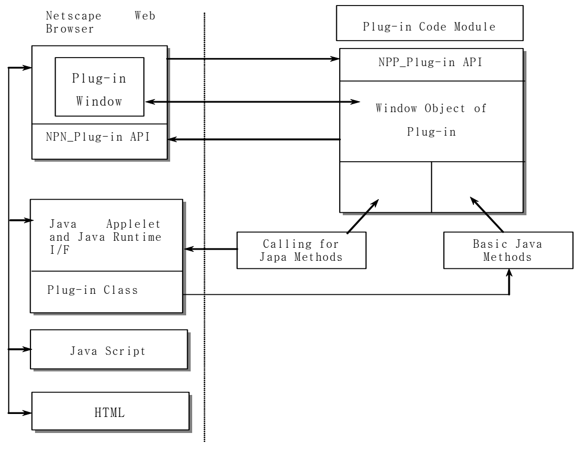

Figure 1 Plug-in Program Structure

The navigator plug-in structure is composed of the plug-in code module, navigator, Java Applelet, Java Runtime Interface, Java Script, and HTML. The core of plug-in interface is plug-in API. Among these APIs, navigator methods start with NPN_, while plug-in methods start with NPP_. Also, LiveConnect connects Java and Java Script to the plug-in.

<LiveConnect>
It is possible to combine Java, Java Script, and plug-ins through navigator LiveConnect. New plug-in methods for LiveConnect are NPP_GetJavaClass, NPN_GetJavaEnv, and NPN_GetJavaPeer. What can be done at LiveConnect are as follows:

Calling Java Method in the Plug-in
Calling Plug-in Methods in Java
Calling Java Method in Java Script
Calling Java Script in Java

<Runtime Loading>
Firstly, confirm code modules in the plug-in directory when performing the navigator for the first time. At this time, the navigator fishes resource information owned by the plug-in. Information related to the plug-in code module can be seen at HelplAbout of the Navigator.

3. Design of the Protective Module

3.1 Structure of the protective module system

The module described in the present paper has been designed and developed in order to guarantee safe transmission of data in the web environment. Particularly, the module designing and implementing cryptographic functions operating in the client node in the web environment is concentratedly dealt with in this paper. In case of sending and receiving data in the web environment, information flows through Internet. There occurs a serious problem if important materials flowing through Internet flow into a third party. Even if the material flows into a third party, it is important to prevent this material from being seen by a third party at least.

Basically, the web environment is a client-server environment. The party offering materials in the client-server environment is the server computer, while the party using these materials is the client system. Data are stored at the server computer, which assumes a role of sending these data to the client as needed. Materials are implemented in the server using HTML most of the time in the web environment. If the quantity of materials is considerable in storing materials using HTML, they are implemented in files using CGI [2, 3, 4, 5].

The present paper illustrates programs which operate based on the module for satisfying the cryptographic function in the client-server structure of the web environment. The server described in this paper has a little different functions from those provided by existing web servers in their concept. The web is open to everyone, however, the server mentioned here is a web server having cryptographic functions to which access is allowed only for a specific person or group. Also, whereas existing web servers have used methods of controlling access in terms of software for cryptographic functions, access is controlled using protective modules having cryptographic functions in the server of this paper. Control of access using protective modules is far stronger than that in the software method. Anyone who does not have a protective module is not able to access the web server from the beginning. In other words, in the existing software method, it is not possible to access without a software program, but it may be possible to access if accessible software is obtained.

Information on the web page has the same form as general documents have. This document is in the form of a file of general type and uses cryptographic functions of the protective module. API interface of the protective module is used in order to use cryptographic functions owned by the protective module. The contents of materials in the form of a general document are not shown directly on the web page in the form of general text or image, but appear on the screen after interconversion of materials is completed using the protective module. This material in a general document form uses CGI (Common Gateway Interface) program module in order to use the protective module. The CGI program is made in order to provide functions which cannot be resolved in terms of HTML language. The CGI program is operated automatically if it is connected from the client to the server, and the contents of a general document are converted to a cryptographic document using cryptographic functions of the protective module through its API. This cryptographed document is then delivered to an accessible client through Internet.

The client to which the cryptographed document is delivered uses the plug-in module in order to decode the cryptographic document. The plug-in module is a technique used in the Netscape web browser. The plug-in module has to decode the cryptographed document after receiving it from the web server. The plug-in module calls API of the protective module in order to decode. API of the protective module is an interface for performing the decoding function of the protective module. The Netscape navigate web browser over which the cryptographed document is handed lets the module decoding the cryptographed document perform by using the plug-in function. Users of the client can view the contents of a normal document through the decoding module. What is described in the above is shown in Figure 2 [10, 11, 12].

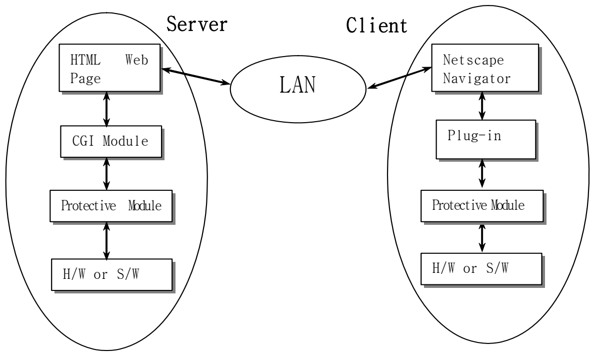

Figure 2 Data transmission model from the server to client

The circumstance shown in Figure 2 corresponds to the case of transmitting data in one direction from the web server to client. Pages in most web servers handle a function of offering information to the client under this circumstance. Most of the time, data are received by the server when we navigate web through the network, but sometimes data are transmitted from the client to server. For example, inputting private information of a user on the web page or entering words on a bulletin board corresponds to this case. A bulletin board may be seen by anyone, but important data such as personal or financial information (for example, credit card number) should not be exposed to anyone. Therefore, in order to solve the above problem, it is possible to stop flowing-out of important personal information by transmitting the client user information to web server by using the protective module.

4. PLUG-IN

4.1 PLUG-IN Registration

The client Netscape navigator finds the program module which is registered in the proper plug-in in the directory at which the Netscape navigator is installed according to the type of document information read by the CGI file offered by HTML of the server when it is connected from the client to Netscape navigator. The plug-in executed is included in the Netscape navigate web browser or separated and shown in the window. When the tex-type document is cryptographed in the server and sent to the client, the client calls the plug-in which decodes the cryptographed document instead of showing it as is.

Firstly, the plug-in should be installed within the directory at which the Netscape navigate web browser is installed in order to enable the plug-in module to be executed. The plug-in has to be installed at the directory of the Netscape navigate web browser. Within this directory, Nppcm.dll plug-in and the protective module for performing the cryptographic and decoding functions have to be installed. The protective module which performs cryptographic and decoding functions is used while Nppcm.dll plug-in module is operated.

After the plug-in module is copied into the Netscape navigate web browser, whether it is correctly registered should be confirmed. The method of confirmation is to execute "About Plug-ins" in the help menu while performing the Netscape web browser.

4.2 PLUG-IN Operation

How the plug-in module is shown in the Netscape web browser of the client after receiving the cryptographic document from the server is illustrated below. When making a web page in the server, in order for the Netscape web browser to call the plug-in in the client, <EMBED> field has to be added to the language. Examples of simple forms of <EMBED> are as follows:

<EMBED src = “output.tex” WIDTH = 100 HEIGHT = 100>

where src is a file for sending the cryptographic document to the web browser of the client. Expanded name (tex) of the file should be written accurately. The expanded name of this file assumes a role of executing the plug-in for the Netscape web browser of the client. Of course, the tex plug-in type should be registered in the Netscape web browser of the client. The next variables are shown to be WIDTH and HEIGHT which mean the width and height, respectively. That is, it is designed that the Netscape web browser calls the plug-in module having the width and height of 100 each in the tex file form.
How the plug-in is executed in the client once a web page in which the plug-in is operated is made in the server is described next.

There is a link for reading the corresponding information in the web page. Clicking of this link allows meeting <EMBED> field with the web page of the server. As illustrated in the above, the plug-in module is called in the Netscape web browser of the proper client according to the file type defined in src file declared in the <EMBED> field when the <EMBED> field is met.

Letters indicating plug-in reading appear briefly and disappear in the state bar when the plug-in is called. Cryptographed materials handed over from the server are then decoded through execution of the plug-in module. After that, information is shown on the Netscape web browser. After the cryptographed document is converted into a document in the general form by performing decoding process of the protective module until information is shown, the screen is then outputted.

5. Conclusion

The present paper is related to guaranteeing safety from a third party when data are sent and received in the web environment. The results of research enable solution of security problems that are negative effects of recent increased use of Internet. The results of study in this paper can be applied to all sites into which important information is implemented in web pages in any organization such as Government, company, etc. Information implemented in web pages can be accessed by non-specific multiples. Users not having a protective module in a specific organization is not only unable to access the web page but also unable to see the material although they obtain the cryptographed material since its decoding is not possible. Generalization of web pages prompts rapid increase in the number of associations, companies, and Government organizations who need such environment. The present paper deals with implementation of the protective function which is necessary in the client system for implementation of the protective system on Internet web pages. The cryptographic function in the client system uses the protective module. The plug-in program function has to be used in order to operate the protective module in the client system. The web protective program module developed through research and study described in this paper is composed of:

References

[1] Zan Oliphant, "Netscape plug-in programming," Info Book, June 1997.
[2] Ed Tittel, Mark Gaither, Sebastian Hassinger & Mike Erwin, "CGI Bible," Young-Jin Publishing Company, August 1997.
[3] Stephen Asbury, "CGI HOW-TO," Dae-Rim, April 1997.
[4] Dwight & Niles, "CGI programming leart from examples," Info Book, May 1997.
[5] Charles Petzold, "Programming Windows 95," Kyohak-Sa, October 1996.
[6] Sang-Yup Lee, "Visual C++ Programming Bible Ver 5.x," Young-Jin Publishing Company, August 1997.
[7] Michael Morrison et al., "Java Unleashed," Dae-Rim, July 1996.
[8] Suk-Joo Kim, "Temptation of Java Script," Kanam-Sa, October 1996.
[9] Cornell, Horstmann, "Core Java," Young-Han Publishing Company, September 1997.
[10] Peter D. Hipson, "Window NT Server," Cyber Publishing Company, March 1997.
[11] Kyung-Man Kim, "Let's implement the web server with IIS," Information Era, April 1997.
[12] Peter Dyson, "INSIDE SECRETS IIS 4," Triangle, May 1998.

Seung-Ju, Jang received a B.Sc. degree in Computer Science and Statistics, and M.Sc. degree, and his Ph.D. in Computer Engineering, all from Busan National University, in 1985, 1991, and 1996, respectively. He is a member of IEEE and ACM. He has been Professor in the Department of Computer Engineering at Dongeui University since 1996. He was a member of ETRI(Electronic and Telecommunication Research Institute) in Daejon, Korea, from 1987 to 1996, and developed the National Administration Multiprocessor Minicomputer during those years. His current research interests include fault-tolerant computing systems, distributed systems in the UNIX Operating Systems, multimedia operating systems, security system, and parallel algorithms.