Design of Web Based Secure Embedded System in C

Bo Qu
School of Mathematics and Information Technology
Nanjing Xiaozhuang University
Nanjing, China
e-mail: Mr.QuBo@126.com

Zhongxue Yang
School of Mathematics and Information Technology
Nanjing Xiaozhuang University
Nanjing, China
e-mail: young-sir@vip.sina.com

***Abstract*—This paper describes the design and implementation of Web based secure embedded system, including tiny secure Web server, embedded system-based CGI library, and utilities for security, such as CAPTCHA program and mail sending program. The entire system is designed in C by the author of this paper on Linux platform with GNU tool chain. A Web based secure embedded application demo system is also designed to show the architecture and running effect. On the premise of implementing essential functions, the scale of the system is designed as small as possible therefore it is suitable not only for embedded Linux development on Web based applications but also for embedded system course teaching in colleges and universities. By some further extensions, it can also be used on development of general Web based MIS.**

Keywords-Web based embedded system; secure Web server; CGI library; CAPTCHA; CGI programming

I. Introduction

With the rapid developments of computer technologies, embedded systems are used almost at anywhere, and B/S (Browser/Server) mode has become a standard form for users to control or access embedded application systems. That is analogous very much to the general computer application systems based on Web such as Web surfing and MIS (Management Information System), etc., which are generally supported by powerful Web servers, such as IIS, Apache, and Databases, such as Sybase, MySQL, etc.

For embedded systems, however, confined by its hardware scale and running cost, its resources of hardware and software are both far less than that of general computer systems. That obviously brings some special issues, for example, the memory space is very small, flash memory instead of ordinary hard disk is used as permanent storage, the operating system is scaled as small as possible therefore lots of functions are omitted, etc.

How to develop required application systems on such small scale embedded systems and, at same time, ensure the security and reliability as far as possible has become a challenge encountered by embedded system developers. The Web based secure embedded application demo system described in this paper gives a valuable example to develop secure applications on embedded systems. The main contributions of it are as following:

- Tiny SSL protocol-based secure Web server. By the reasons above, general purposes Web servers, such as well known Apache, are not suitable for embedded systems, and in fact, there is as yet no acknowledged one. The tiny secure Web server described in this paper is just a solution for this issue, which can not only implement the secure Web service reliably, but also save the memory space of the embedded system effectively.
- Embedded system-based CGI library. Considering the feature of embedded systems, only foundational and necessary CGI functions are realized, for example, getting forms, query strings and cookies. However, that does not means its capacity is deduced since other more functions can be obtain by simple extensions based on these basic functions.
- Utilities for security. To improve the security of corresponding embedded application systems, some useful utility functions are realized, for example, image CAPTCHA (verification code) function for secure login, mail sending function for getting back lost password, etc.

This paper describes the design and implementation of Web based secure embedded application system in C [4], and presents a demo example, developed on Linux platform with GNU tool chain, to show the running effect on an actual embedded platform.

II. Tiny Secure Web Server

Secure Web server is the basis of a Web based secure embedded system. With HTTP protocol, however, security and confidentiality of communicating data is not available since it uses plaintext to communicate between clients and servers. A popular way is using HTTPS [1] protocol which realizes the encryption of data communication with the mechanism of Secure Socket Layer (SSL) [2] protocol.

The author of this paper has designed such a secure Web server for ARM platform and published a paper [5] on it. For the system described in this paper, the program is reduced further while the CGI manipulating functions are improved, of which the amount of code is 500 lines and the size of the executable file is 25K only, obviously small enough. The server consists of three parts: SSL manipulation, HTML file processing and CGI execution.

SSL manipulations include initialization of SSL, reading and writing with SSL, as well as creation and identification of SSL certificate. To use SSL protocol, the corresponding SSL link library, openssl, must be ported to embedded system. For the communications between clients (user browsers) and server, the SSL protocol is used while the data transmission between server and CGI programs uses normal pipes without SSL.

HTML file processing is simplified to accomplish only relatively simple functions. It can identify commonly used file types, such as audio files (mp3, mpa, abs, mpega, wav), video files (mpeg, mpg, mpe, mpv, vbs, mpeg), application files (bin, com, dll, exe, tar, zip, doc, xls, ppt), image files (bmp, gif, jpg, jpeg, png), and text files (txt, htm, html), etc.

CGI execution is the most important component of the server. Some necessary environment variables are provided by the server for CGI programs, of which the most important ones are "REQUEST_METHOD", "CONTENT_LENGTH", "CONTENT_TYPE", "HTTP_COOKIE", etc. To transmit data between server and CGI programs, two pipes are established of which one for server writing to CGI and another reading from it.

The implementation details are described in [5] therefore no further mention is provided here.

III. EMBEDDED SYSTEM-BASED CGI LIBRARY

As we all know, some Web authoring languages, such as PHP, ASP, JSP, etc., are popularly used in Web based application systems especially in recent years. Although using these languages can improve the programming efficiency to a certain extent, specially designed dedicated Web servers, however, are required to support these languages, for example, Apache for PHP, IIS for ASP, etc. JSP is based on Java environment therefore at least a Java virtual machine is needed. Obviously, most of embedded systems can't meet such requirements.

CGI (Common Gateway Interface) [7] is a specification for Web service with which the Web server can execute scripts or binary programs to accomplish some dedicate tasks unable to be done by server itself or ordinary web pages. Its best advantage is that it needs not any particular functions for Web server, in other words, CGI programs can be invoked by any standard Web server directly. Considering the characteristics of embedded systems and the capability of C language, obviously, designing CGI programs in C for embedded system applications is the best choice.

A. Overall Structure of CGI Functions

According to HTTP protocol, the information transported between client and server can be divided into three types: query strings, e.g., that appended on the tail of a URL started by "?", forms, e.g., that used in a login page, and cookies, defined by server while stored in client. For each type, there can be multiple entries with the form: name=value, e.g., in the following query string there are two entries:

http://x.com?name=abc&page=2

in which the first entry is "name" with the value of "abc" and the second is "page" with the value of "2".

As we all know, by using Web service with HTTP protocol, information can be not only downloaded from the server to client, but also uploaded from client to server. Further more, files stored in client host can also be uploaded to server by the protocol. The CGI library mentioned in this paper supports both downloading and uploading functions.

Since information is encoded when it is transported through networks, it is needed to be decoded back into original. That is not complex therefore not described further confined to the length of the paper.

B. Getting Information Functions for CGI

For a client query page, there are two important types of query methods: GET and POST. According to HTTP protocol, the ways of them to provide information is different therefore dedicated functions must be defined to manipulate each of them. While on the other hand, the cookies are provided by client in the message header and converted by Web server to environment variables. In the CGI library, function _decode() is defined to decode the three types of information and three pointers are defined to point to them.

For query method GET, the query string is pointed to by environment variable QUERY_STRING, and, since it is an ASCIIZ (terminated with ASCII 0) string, the length of it can be obtained by using function strlen().

For query method POST, the length of the information is stored by the Web server in environment variable "CONTEN_LENGTH", while the content of it is transmitted from Web server through standard output.

For cookies, the cookie string is pointed to by environment variable "HTTP_COOKIE", and, it is also an ASCIIZ string.

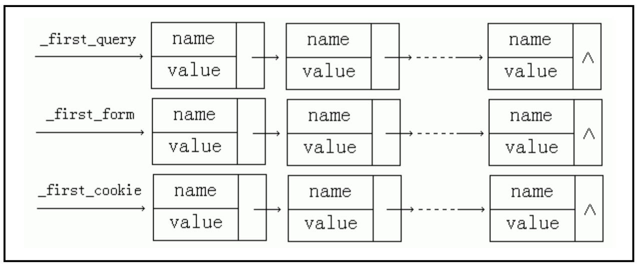


Figure 1. Linked list of query, form and cookie entries

Since the three types of information are all in the form as:

name0=value0&name1=value1&…&namen=valuen,

a function, named _make_entry(), is designed to decompose the string into a linked list as shown in Fig 1. By this mechanism, the entries in different kinds of list with a same name can be identified as different entries therefore more convenience will be provided for programming.

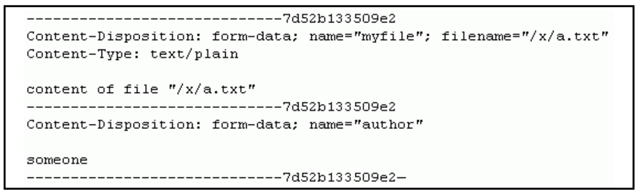
Corresponding to the three linked lists, the CGI library provides three interface functions, i.e., cgi_query(), cgi_form() and cgi_cookie(), respectively. For convenience to manipulate numerical information, another three corresponding functions are also designed to return integer values, i.e., cgi_iquery(), cgi_iform() and cgi_icookie().

C. Manipulating File Uploading

Originally, there is no file uploading function in HTTP protocol before it is added by RFC1867 [6]. With this specification, clients such as IE, Mozila, Opera, etc., can transmit files designated by users to server. The popularly used Web authoring languages, such as PHP, ASP, JSP, etc., can all implement the functions of manipulating file uploading therefore can resolve the transmitted user file with the specification. But for CGI programs in C, all CGI functions including file uploading functions must be manipulated by programmer him or her self.

Based on original HTTP protocol, RFC1867 adds a new attribute FILE to TYPE attribution of INPUT tag, and at the same time requires the METHOD attribution must be “POST” and ENCTYPE be “multipart/form-data”. The main changes by RFC1867 are as following.

- Change to HTTP message header: Content-Type line is changed from “Content-Type: application/x-www-form-urlencoded” to, for example, “Content-Type: multipart/form-data; boundary=-------------------------7d52b133509e2”, that is, adds a “boundary” to delimitate the entries of HTTP, in which the numeric characters are generated randomly.
- Change to HTTP message body: Now that there are both normal entries and file entries to be uploaded, all of them are delimitated by boundaries. An HTTP entry may appear as in Fig 2.



```
-----------------------------7d52b133509e2
Content-Disposition: form-data; name="myfile"; filename="/x/a.txt"
Content-Type: text/plain

content of file "/x/a.txt"
-----------------------------7d52b133509e2
Content-Disposition: form-data; name="author"

someone
-----------------------------7d52b133509e2-
```

Figure 2. An example of HTTP body with file uploading

In Fig.2, the first line describes the file to be uploaded, in which, “Content-Disposition:” means the type is “form-data”, “name=myfile” sets the name of <INPUT type=”file”> as “myfile”, and “filename=/x/a.txt” demonstrates the full path of the uploaded file in client. The next empty line represents the end of the message header, therefore the content of the uploaded file comes next. After the content of the uploaded file, the boundary comes again representing the end of the first HTTP entry and the beginning of the second if there exists, in this example, the text entry named “author”. Finally, another boundary appears with the postfix “--” representing the whole HTTP message body is ended. According to the structure of Fig. 2, the corresponding manipulating routine will not be difficult to be programmed.

An important issue for file uploading is that how data to be buffered during uploading. There are two ways to deal with this issue, i.e., by temporary file or by temporary memory. For general purposes computer systems, the volume of hard disks is big enough therefore by temporary file is a good choice. For ordinary embedded systems, however, since most of them use flash memory instead of disks as secondary storage which is not suitable for frequently writing, let alone the volume of it is relatively smaller. That means the temporary memory should be used as buffer for file uploading. The CGI library mentioned in this paper uses just this way.

To store the uploaded file into file system, an interface function is designed, the prototype of which is as following:

int cgi_loadfile(char *file, char *buff, int size);

in which “buff” is the temporary buffer, “size” is the length of the uploaded file, and “file” is the path name of destination file to be saved on the server’s file system. The function will create a new file named by “file”, and then copy the content stored in the buffer “buff” into the file.

D. Setting Cookies

Setting cookies is an important function for CGI since cookies is almost an inevitable component when user identification is needed. According to HTTP protocol, the lines of setting cookies must appear within the acknowledge message header from Web server.

A cookie element can have several attributions, i.e., name, value, expire time, domain, path, security, of which, the first two is necessary representing the name and value of the cookie respectively, the third means the time expiring it the cookie will be invalidate, the forth and fifth describes the domain and path at which the cookie is valid, and last means only by secure method can the cookie be transmitted if it is set to “secure” and the default setting is by non-secure ways.

In the CGI library, the cookie setting function cgi_setcookie() can manipulate all the six attributions but only the first two are necessary. That means the latter four can be set to 0 or NULL representing “no setting”.

E. Miscellaneous Functions

Some other commonly used functions for CGI are also designed, such as cgi_redirect() to redirect to another URL, cgi_downloadmime() to create a message header for a request page to download a file, cgi_URLencode() and cgi_URLdecode() to encode or decode a URL respectively, and cgi_chkemail() to check syntax of an email address, etc.

Due to the features of embedded systems, the tiny CGI library mentioned in this paper is designed elaborately, of which the amount of source code is only 580 lines and the size of the binary library is only 19K. It is shown by practice that it can indeed meet the general needs for ordinary embedded application systems.

IV. UTILITIES FOR SECURITY

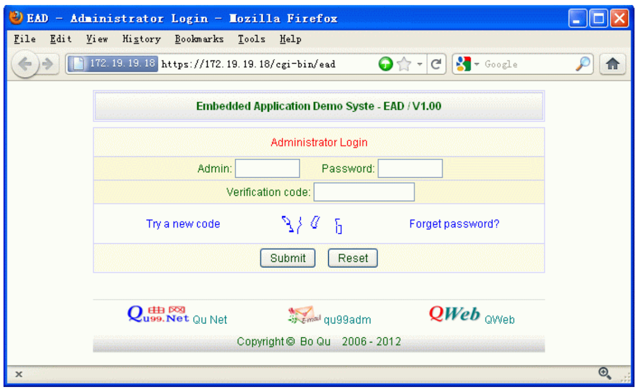
As we know, an ordinary Web based login system involves at least a user name and password. Only if both of them are verified to be correct by the Web server, can the user enter the system. Since the well known HTTP protocol is used, however, such systems are easily attacked by malicious users with some so called auto password detectors. Obviously the security of the system will be debased. To remedy the defect, a popularly used method is by using verification code during login procedure. By this way, not only the right user name and password are required but also the correct verification code is needed. This is just the technique known as CAPTCHA [3], standing for "Completely Automated Public Turing test to tell Computers and Humans Apart". Although some of powerful graphics libraries are exist to support the design of such image form CAPTCHA, most of the embedded systems are not able to install them because of the limited resources. That means the CAPTCAH programs in ordinary embedded systems should be independent of any graphics library. By this reason, a tiny but practical CAPTCHA program is designed in C by the author, which can form an image consisting of a numeric string, without the supporting of any graphics library.

For the purpose of getting back lost password, a simple mail sending program is also designed according to the SMTP specification [8]. By using the program, the new password of the user who lost his or her password will be sent back to the user's mail box.

Due to limited space of the thesis, the technical details of implementing these utilities are not mentioned furhter.

V. A DEMO EXAMPLE

In order to show the effect of the system, a small embedded application demo system is designed by the author, called EAD for short. It is typically a CGI program designed by using the embedded system-based CGI library mentioned in this paper. The Web server used is the tiny secure Web server mentioned also in the paper. The testing platform is an embedded development board, named FL2440, with arm-linux platform.


EAD - Administrator Login - Mozilla Firefox

File Edit View History Bookmarks Tools Help

172.19.19.18 https://172.19.19.18/cgi-bin/ead

Embedded Application Demo Syste - EAD / V1.00

Administrator Login

Admin: Password:

Verification code:

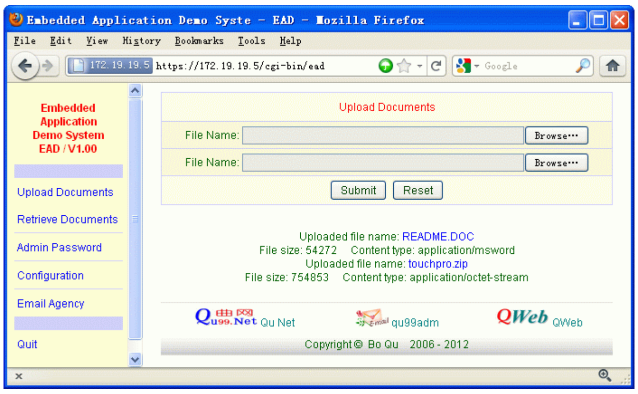
Try a new code Forget password?

Submit Reset

Qu99.Net Qu Net qu99adm QWeb QWeb

Copyright © Bo Qu 2006 - 2012

Figure 3. Login page of EAD


Embedded Application Demo Syste - EAD - Mozilla Firefox

File Edit View History Bookmarks Tools Help

172.19.19.5 https://172.19.19.5/cgi-bin/ead

Embedded Application Demo System EAD / V1.00

Upload Documents

Retrieve Documents

Admin Password

Configuration

Email Agency

Quit

Upload Documents

File Name: Browse…

File Name: Browse…

Submit Reset

Uploaded file name: README.DOC
File size: 54272 Content type: application/msword
Uploaded file name: touchpro.zip
File size: 754853 Content type: application/octet-stream

Qu99.Net Qu Net qu99adm QWeb QWeb

Copyright © Bo Qu 2006 - 2012

Figure 4. Demo of uploading files

Fig. 3 shows the login page of EAD. Note that the protocol used in the URL is "https" which means the secure HTTP is used. The numeric characters, "9106", in the CAPTCHA image are obviously distorted which will increase the difficulty greatly for automatic detectors. The super link "Try a new code" is used to refresh only the verification code image itself without requiring reloading the entire page. The super link "Forget password?" is used to get back the lost password in which case a new password will be set by EAD and mailed to the administrator's mail box automatically.

Fig. 4 gives an example of uploading files. In this case, two files are uploaded together to the embedded server and the names, sizes and content types of the files are shown in the page. By clicking a file name on the page, which is a super link, the just uploaded file can be downloaded to the client by the browser. That means the files are indeed uploaded correctly.

VI. CONCLUSION

It is can be said that information security is an eternal topic. With embedded application systems being popularly used more and more in a wide variety of different fields, from tiny remote control systems to smart mobile phones, the requirements of ensuring the embedded systems more secure and reliable are also intense more and more. The descriptions given in this paper present a valuable example to design and implement such a Web based secure embedded application system.

Essentially, the techniques mentioned in this paper not only can be used in embedded application system designs, but also is suitable for general Web based application systems such as MIS, etc., as well as embedded system course teaching in colleges and universities. Therefore, strengthening the research on the field is undoubtedly of great theoretical and practical values.

REFERENCES

[1] A. S. Tanenbaum, "Computer Networks, Fourth Edition", Prentice Hall, Inc.,2008.

[2] W. Stallings, "Network Security Essentials: Applications and Standards, Third Edition", Prentice Hall, Inc., 2007.

[3] L. V. Ahn, M. B. and J. Langford, "Telling Humans and Computers Apart Automatically", Communications of the ACM, volume 47, pp. 57-60, 2004.

[4] M. J. Rochkind, "Advanced UNIX Programming", Second Edition. Addison-Wesley, 2006.

[5] B. Qu and Z. Wu, "Design and implementation of embedded secure web server for ARM platform", Proceedings of 2011 International Conference on Electronic and Mechanical Engineering and Information Technology, EMEIT 2011, volume 1, pp. 359-362 , 2011.

[6] E. Nebel and L. Masinter, "Form-based File Upload in HTML", http://www.ietf.org/rfc/rfc1867.txt, 1995.

[7] D. Robinson and K. Coar, "The Common Gateway Interface (CGI) Version 1.1" , http://www.ietf.org/rfc/rfc3875.txt, 2004.

[8] J. B. Postel, "Simple Mail Transfer Protocol", http://www.ietf.org/rfc/rfc0821.txt, 1982.