Insecurity of Inputs to CGI Program

By Suhairi Mohd Jawi @ Said

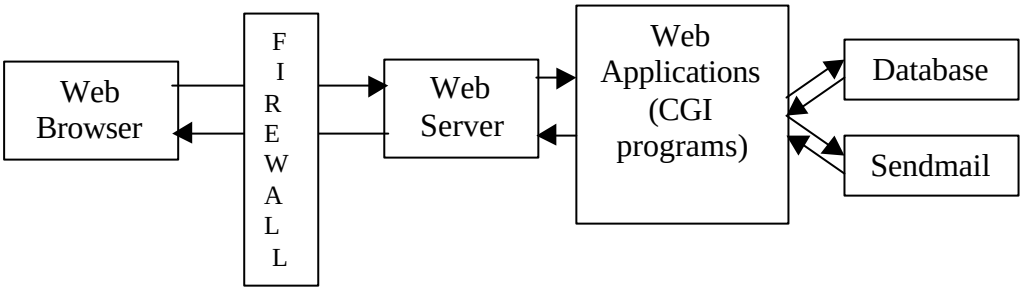
Table of Contents

Purposes

This paper is to list some points that each web programmer has to consider while coding a web based application that interacts with user inputs through CGI as well as tools that can be used to test it.

Introduction

Common Gateway Interface or *CGI* is a method for web browser (client) to interact with host operating system through a web server. CGI allows the client to run a program or web application on the host machine. The program can be written in any programming language whether it is compiled or interpreted, as long as it is executable and written correctly. The following figure shows the components of the client, server and CGI program.



The program usually will interact with other applications or services on the operating system of the web server to complete the tasks. They can be database server, mail program and content services. The program will pass the user inputs (obtained from the URL or HTML form) and states (from cookies, hidden fields and environment variables) to the application to be processed and the results will be passed back to the client.

One of the drawbacks of this technique is mostly concerned with user inputs and the states of the transaction. There is no guarantee that a user will enter the input correctly (intentionally or accidentally) and states (values of cookies, hidden fields and environment variables) may be modified.

Following are the discussions of some known issues regarding inputs while developing and implementing the CGI program. At the end of this article, a tool called RFProxy by Rain Forest Puppy will be discussed that can be used to test a CGI program against user inputs.

1. Buffer Overflow of Faulty CGI Source Code

Buffer overflow is a well-known attack where the length of the user input is greater than maximum length of input accepted by the program. The overflow part means that after the maximum number of characters, the input become executable code that is used to take over or crashes the system.

Consider the following code:

```
#include <stdio.h>
#gcc test.c –o test
int main(int argc, char *argv[])
{
        char user_input[255];
        puts ("Content-Type: text/html\r\n\r\n");
        while (argc--) {
                strcpy(user_input, argv[argc]);
                puts(user_input);
        }
        return 0;
}
```

and this request:

```
http://url/cgi-bin/test?AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA…AAAAA
```

where AAA part is longer than 255 characters. The program *test* program just simply crashed. However, there are some severe results on some system. An expert user may be able to craft a string of characters that can be used to overwrite the program instruction pointer which points to the next program instruction. If the overflow part contains a valid memory address, the next program instruction will not be executed. Instead, the code, which is pointed by the new instruction pointer, will be executed. This can be used to call another program routine or to avoid security check.

This type of input may be passed from the encoded URL as above or through a form field. In order to minimize or stop this threat, the author must determine the length of the legitimate input.

At the user level in the form fields of type TEXT and PASSWORD, there is an option to specify the MAXLENGTH of the character input. However, this will not guarantee the length of input is secured. Any user may edit the form offline or use some tools such as interception proxy that can modify the MAXLENGTH before send it to the server.

At the application level, depending on the functionality of the program, the author can stop the processing and print error messages to the user with some warning. There are some ways if the execution is allowed to proceed:

(a) Array bound must be checked before write it to the buffer. Or
(b) Only use string function that can specify explicit buffer size.

2. Unfiltered Metacharacters

Metacharacters are characters other than alphanumeric. These special characters may carry some meaning and functionality in some programming languages, interpreters, applications and operating systems. If these characters are passed without checking, it may cause some undesirable results and disasters.

These characters are: ; < > * | ‘ & $! ( ) [ ] { } : ‘ “ / #

For examples,

(a) ‘<’, ‘>’, ‘“’ and ‘&’ can affect the rendering of the html in the browser.
(b) In perl, $ and & used as prefix to variables and function calls respectively.
(c) In some web server, combination of ‘/’ and ‘.’ can be used to traverse directories.

One solution to these is by converting them to other representations. Many perl programs have a function call to convert them into HTML special characters code, such as

```
sub htmlify {

      my($string) = shift;

      # Special Characters; RFC 1866
      $string =~ s/&/&amp;/g;
      $string =~ s/\"/&quot;/g;
      $string =~ s/</&lt;/g;
      $string =~ s/>/&gt;/g;
      .  .  .
      return $string;
}
```

The complete references of HTML conversion can be obtained at http://www.bbsinc.com/iso8859.html. Other solution is by inserting an escape character (\) before the metacharacter in the code so it will not be evaluated. However, to ensure total security of the web server’s host and the underlying application, it is necessary to wipe them out completely.

Filtering metacharacters are important especially for application which

(a) Writing Server Side Include (SSI) tag to a SHTML enabled page such as guestbook application. Tag like:

```
<!--#exec cmd="/bin/rm -rf /"-->
```

will delete everything on the machine of web server. This can be avoided by disabling `exec` functionality in the web server.

(b) Passing user inputs as arguments to a system or shell command of the operating system of the web server. For example, system command can be invoked in perl script by using `system(arguments)` and `exec(arguments)`.
(c) Forward request to another application such as database server and mail server.
(d) Creating a file on the fly. Attacker may be able to create a .cgi file and execute it remotely.

3. Invalid States

The stateless characteristic of the HTML makes it not interactive. Any web server does not have any clue what are the previous states of the transaction between client and server. In order to overcome this, many programmers manipulate cookies, hidden fields and environment variables in the program codes as an integrity check of the transaction status.

(a) Cookies

A cookie is a small file stored on the client side. Programmers use cookies to store user information such login and account information, browsing history and transaction information. Unfortunately, cookies are easily exploited and modified. Someone may change the values and then application may get false information about the users. One good example of this is the incident happened to Yahoo and Hotmail (http://www.nbnn.com/news/01/166436.html) where someone could open other email accounts by stealing the cookie values.

(b) Hidden Fields

Hidden field is a type of HTML form input. Some programmers usually store some values to track the transaction status and the next action to be taken by the target application. Like cookies, values of hidden fields can be modified by editing the form offline or online. Therefore, some confidential information should not be included in the code such login, password, application server details and email address.

(c) Environment Variables

Environment variables are values passed by web server to the CGI scripts. They carry some information regarding the browser and the server. Listing of these variables is available at http://perlfect.com/articles/cgi_env.shtml. In perl, these variables are contained in the hash `%ENV` while in C they can be retrieved by calling `getenv()`. However, these values can be manipulated by some methods.

All the values carry by the above entities can be manipulated. Some web applications may be waiting for expected input from the client. If the expected input is not arrived or the value of the input is not valid, the application may be in waiting state for some time or just simply crashed and may lead to denial of service because other clients may not be able to make request.

In Black Hat Briefings 2001 in Singapore, Rain Forest Puppy had used a sample shopping cart program in his presentation entitled Web Assesment Tools to demonstrate how a user can manipulate the above values. A cookie can be used to store login information but later someone else may be able to login by creating a cookie containing valid user information. While doing shopping, a user can change the price tags and be able to buy a number of items with cheaper prices. The following section will discuss

some features provided by RFProxy to assess the security of CGI programs against user inputs as well as other tools that have same functionalities.

RFProxy – A Web Assessment Tool

RFProxy is written by Rain Forest Puppy and was released at CanSecWest 2001 in Vancouver. It is written in perl and the source code can be obtained from rfp.labs homepage at http://www.wiretrip.net/rfp/. The program is run on the client side to monitor or modify incoming or outgoing HTTP connections. The snapshot of its interface is included in the Appendix A.

Although it is meant for auditing CGI program, it is capable to pose threat to any CGI programs that does not have proper implementation and robustness. Among features that is related with the above discussion on input to CGI program are:

(a) HTML rewriting enable

When a client requests a page from a server, the requested page is intercepted and rewritten. All input type values in the html form are made visible, well indicated and modifiable as shown in the Appendix B. For example:

- TEXT and PASSWORD
  - Input type and form name can be viewed by clicking the green triangle
  - Characters are made visible during typing in password field.
  - Input length are not limited therefore MAXLENGTH becomes useless.

- HIDDEN field are not hidden anymore and value can be changed.

- Values for OPTION in SELECT field can be changed.

  This capability can test how a CGI program will behave when it receive inputs with values that never been tested. Modified inputs may cause buffer overflow, arbitrary command execution or crashing the program.

(b) Record requests for later replay

The communication between the client and server is recorded. User can replay the entire session or a portion of it at some later point in time. This can be used to test a program that manipulates session id or examine the pattern of requests submitted by the client during communication.

(c) Intercept and handle cookies

Sites that use cookies for authentication in order to store some details after a successful login are vulnerable to the changing cookies values. An attacker may guess or steal values of a cookie in order to get access into somebody else account.

(d) Override 'Referer' header with new value

Referer is a value contained in environment variables. It tells a CGI program from which URL the current page is linked. It can be used to validate whether a user make request from a valid site such as after authentication. If the attacker changes the Referer value, he may gain unauthorized access to the site.

Other Web Assessment Tools

There are some other tools that can be used to test CGI program against user inputs.

(a) AppScan

Appscan is developed by Sanctum (http://www.sanctuminc.com) and costs with some high price. This can be used to check CGI program against application tampering, cookie poisoning, third-party misconfiguration, known Web vulnerabilities and buffer overflows.

(b) HTTPush

HTTPush's funtionality is almost similar to the RFProxy and it is also written in Perl by Lluis Mora of S21SEC (http://www.s21sec.com/download/httpush-current.tar.gz) and available freely. Its development is farther than PFProxy and also supports SSL.

(c) Achilles

Achilles is available freely at http://www.digizen-security.com/projects.html. It is a proxy server acting as a man-in-the-middle during HTTP session between client and server. The ability to intercept and alter an HTTP session's data can be used to test security of web applications against user inputs.

Conclusion

Most of the web applications available for free are using the same libraries and techniques. Even the new scripts or codes sometimes copy somebody else coding. It seems the above problems may not be easily wiped out if the author does not take care what has been written, copied and reused.

Application developer must learn how to handle input correctly and not give full trust to user input and application since they are all can be tempered. They are always new tools available in the Internet and they can be used to test the CGI program.

List of References

[1] K.1 Information Security Kickstart Highlights,
Programmatic & Web Security, 6.13 – 6.19, 6.23 – 6.24
SANS GIAC © 2000, 2001.

[2] Eric Kim, Eugene. Chapter 9: CGI Security, Writing Secure CGI Programs.
17 December 1996. URL: http://w3rum.upr.clu.edu/htmldocs/cgi/ch09/0903.html

[3] J. Brewer, Kevin. ASCII – ISO 8859-1 (Latin-1) Table with HTML Entity Names.
31 January 1997. URL: http://www.bbsinc.com/iso8859.html

[4] Bartlett, Michael.Vulnerability Discovered In Yahoo Mail, Hotmail. 4 Jun 2001.
URL: http://www.nbnn.com/news/01/166436.html

[5] D. Stein, Lincoln. & N. Stewart, John. The World Wide Web Security FAQ:
CGI Scripts, 28 July 2001. URL: http://www.w3.org/Security/Faq/wwwsf4.html

[6] Moraitakis, N. & Zervas, G. CGI Environmental Variables, 2000.
URL: http://perlfect.com/articles/cgi_env.shtml

[7] Rain Forest Puppy. BlackHat Asia conference materials, Singapore. April 2001
URL: http://www.wiretrip.net/rfp/talks/blackhat-asia-2001/

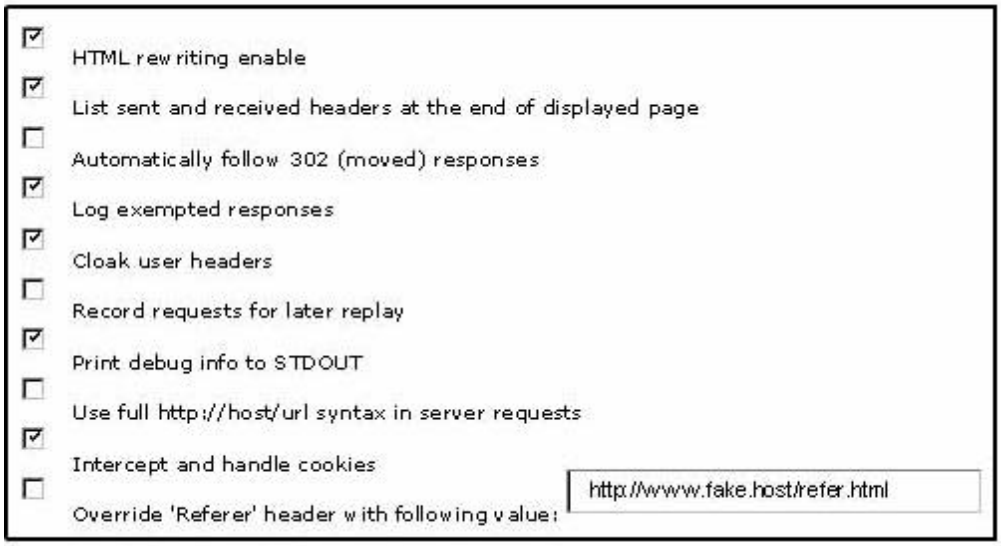
[8] Kalev Ashdod, Danny. Avoiding Buffer Overflows.
URL: http://www.devx.com/free/tips/tipview.asp?content_id=2912

[9] Forristal, Jeff. AppScan Flags Security Problems in Web Applications.
16 October 2001. URL: http://www.networkcomputing.com/1120/1120sp3.html

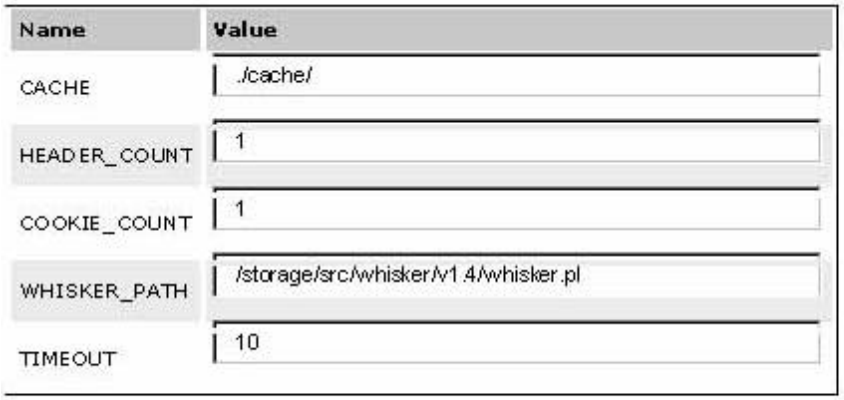
[10] DigiZen Security Group, Achilles, 2000,
URL: http://www.digizen-security.com/projects.html

Appendix A: RFProxy Administration Interface

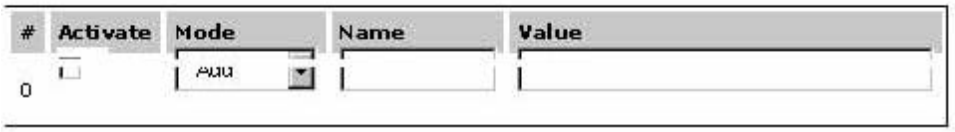
General options:how



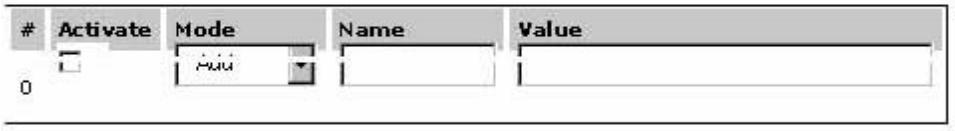
Various configuration values:



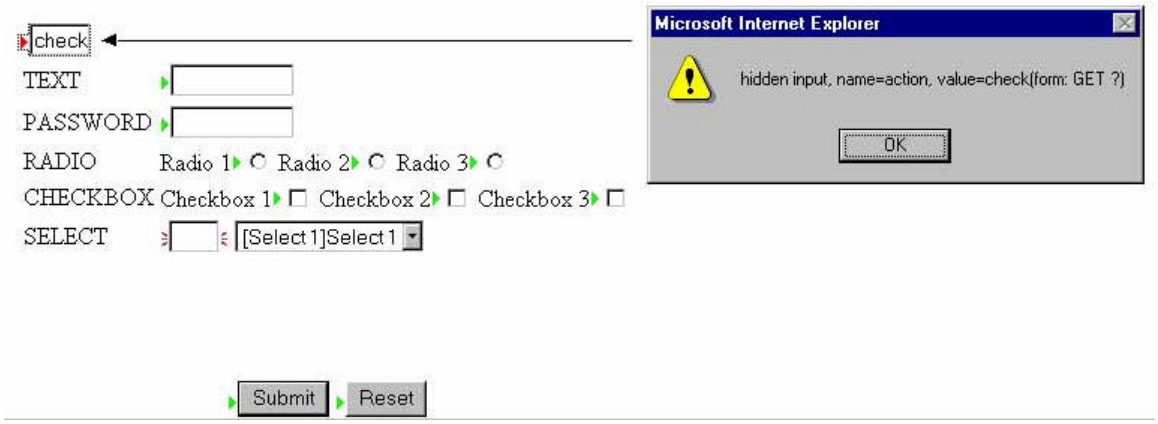
Header modification rules:



Cookie modification rules:



Appendix B: Sample Form Parsed by RFProxy



Original HTML Source Code:

```
<HTML>
<HEAD>
 <TITLE>Test Sample Form</TITLE>
</HEAD>
<BODY>

 <FORM ACTION="?" METHOD=POST>
  <INPUT TYPE=HIDDEN NAME="action" VALUE="check">
  <INPUT TYPE=HIDDEN NAME="username" VALUE="john">

  <TABLE>
   <TR><TD>TEXT</TD><TD><INPUT TYPE=TEXT NAME="text" SIZE=10 MAXLENGTH=10></TD></TR>

   <TR><TD>PASSWORD</TD><TD><INPUT TYPE=PASSWORD NAME="password" SIZE=10
           MAXLENGTH=10></TD></TR>

   <TR><TD>RADIO</TD><TD>
    Radio 1<INPUT TYPE=RADIO NAME="radio" VALUE=1>
    Radio 2<INPUT TYPE=RADIO NAME="radio" VALUE=2>
    Radio 3<INPUT TYPE=RADIO NAME="radio" VALUE=3>
   </TD></TR>

   <TR><TD>CHECKBOX</TD><TD>
    Checkbox 1<INPUT TYPE=CHECKBOX NAME="check" VALUE=1>
    Checkbox 2<INPUT TYPE=CHECKBOX NAME="check" VALUE=2>
    Checkbox 3<INPUT TYPE=CHECKBOX NAME="check" VALUE=3>
   </TD></TR>

   <TR><TD>SELECT</TD><TD>
    <SELECT NAME="select">
     <OPTION VALUE="Select 1">Select 1
     <OPTION VALUE="Select 2">Select 2
     <OPTION VALUE="Select 3">Select 3
    </SELECT>
   </TD></TR>

   <TR><TD COLSPAN=2><BR><BR><BR><BR></TD></TR>

   <TR><TD COLSPAN=2 ALIGN=CENTER>
    <INPUT TYPE=SUBMIT VALUE="Submit">
    <INPUT TYPE=RESET VALUE="Reset">
   </TD></TR>
  </TABLE>
 </FORM>
</BODY>
</HTML>
```