Paper 194

Intranet Security and SAS®: A Brute Force Approach

Kerril Bauerly, LabOne, Inc., Lenexa, KS

Introduction

As the largest insurance laboratory in the United States, we generate literally thousands of pages of reports each reporting period. These reporting periods range from daily to yearly. In order to cut down on some of the paper and to make data more easily accessible, we began work on a company Intranet. The initial idea was to give executives easy access to reports that might otherwise get lost on a cluttered desktop. With that goal in mind, LabOne, LIVE! (**L**aboratory **I**nformation **V**isualization **E**nvironment) was born.

This Intranet system uses the SAS broker and application dispatcher available with SAS/INTRNET software in addition to the SAS HTML formatting macros. Due to the sensitivity of our data, security is a big issue. This paper primarily covers the implementation of our internal security.

The HTML code for our system resides on the network in the directory designated by our Network Administrator. The SAS code also resides on the network in close proximity to the broker. However, since all of our data resides on an Open-VMS Alpha box, we chose to leave it there. We use SAS/SHARE to access it on the Alpha.

Login screen

People with access to our Intranet may reach it by opening their browsers and calling up our login screen. Because we do not have any kind of encryption software available to us, we have had to invent our own security procedures and then implement them.

Html

The chunk of code displayed from LOGON.HTM creates an extremely basic HTML form which allows the user to enter their userid and password and then press enter or click on the submit button. I will note here, that I incorporated a very basic javascript subroutine to make sure both userid and password fields are filled in prior to submission of the form to the broker.

Notice the METHOD=POST option on the FORM tag. The METHOD attribute of the <FORM> tag determines how the form's data is sent to the server for processing. METHOD=GET displays all parameters sent to the broker in the browser address field. METHOD=POST does not. Because it does not display the parameter values, this is the most secure way to pass an unencrypted password to the broker.

LOGON.HTM

```
.
<P>
<FORM NAME="login" METHOD="POST"
ACTION="/cgi-bin/broker.exe" onSubmit="return
validateForm()">
Enter your user name: <BR>
<INPUT  TYPE="TEXT" NAME="userid" SIZE="10"
MAXLENGTH=8 ALIGN=left><BR>
Enter Password: <BR>
<INPUT  TYPE="PASSWORD"  NAME="password"
SIZE="10" MAXLENGTH=8 ALIGN=left><BR>
<BR>
<BR>
<INPUT NAME="submit" TYPE="SUBMIT"
VALUE="SUBMIT" ALIGN=middle><BR>
<INPUT TYPE="HIDDEN" NAME="_PROGRAM"
VALUE="prod.logon.sas">
<INPUT TYPE="HIDDEN" NAME="_SERVICE"
VALUE="prod">
<INPUT TYPE="HIDDEN" NAME="_DEBUG" VALUE="0">
.
.
```

We are using hidden fields to transmit the program name, service name, and debug options. _SERVICE=”prod” tells the SAS application broker the Application Server name. We have test and prod app servers set up to create test and production environments. _PROGRAM=”prod.logon.sas” is the name of the program the broker needs to execute. Prod is the libref of the directory where our production SAS programs reside. (This is defined in SRVAUTO.SAS by the Network Administrator.) _DEBUG= is a SAS variable that lets you turn on varying degrees of debugging. A value of 131 echoes all system and browser variables, the elapsed time, and sends the SAS log to the browser session.

Style sheet

I will note here that we wanted each of our HTML pages to have the same look and feel. To do this we set up a Cascading Style Sheet (CSS) which contained attributes for all the tags we wanted to act the same on every page. Because CSS is really beyond the scope of this paper, I will just say that using it gave each screen the same background and heading attributes. It is invoked by a <LINK> tag in the top of each HTML page in our Intranet system.

We also established a standard heading which can be included in pages or not, depending on the content.

SAS program

The SAS program, LOGON.SAS, is invoked by the Application Broker. Since we store our data up on the Alpha and use SAS/SHARE to access it, we must define our remote connection protocol in the OPTIONS statement. The LIBNAME statements define the necessary directories on the alpha box to the SHARE server.

LOGON.SAS Excerpt 1

```
OPTIONS COMAMID=TCP MPRINT;

/* declare remote libraries */
LIBNAME PRIVATE 'SAS_DATA1:[SAS_DATA.TEST]'
SERVER=SAS.SAS_RMT;
LIBNAME SASTEMP 'SAS_TEMP:'
SERVER=SAS.SAS_RMT;

/* declare variables coming from web page */
%GLOBAL USERID PASSWORD;

PROC SORT DATA=PRIVATE.APPLFUNC(READ=WEBS)
OUT=ONE;
  BY USERID;
RUN;

/* lookup userid and password in applfunc
dataset */
PROC SQL NOPRINT;
  SELECT USERID, PASSWORD, APP_CODE
  FROM ONE
  WHERE UPCASE(USERID) =
        UPCASE(SYMGET('USERID')) AND
        UPCASE(PASSWORD) =
        UPCASE(SYMGET('PASSWORD')) AND
        APP_CODE = 'SASWEB';
QUIT;
```

We use PROC SQL to ensure the user trying to log on has entered both a valid userid and password. We also track beginning and ending dates, so it is possible for a user's access to expire. We use this feature primarily to turn users on and off without deleting them completely.

Applfunc dataset

Our APPLication FUNCtion dataset contains all valid users for our Intranet Application, their corresponding passwords and other fields necessary to determine user access.

Because we wanted to make our INTRANET as secure as possible, we make users change passwords every 30 days. This necessitated the PWDATE field in the APPLFUNC dataset. Users can also change a password on an ad hoc basis.

```
Libref: DATATEST
Dataset: APPLFUNC

 Variable Length     Key Label

 USERID   $8          N  LOGON USER ID
 APP_CODE $6          N  APPLICATION CODE
 EFFBEGIN $8          N  EFFCTV BEGIN DATE
 EFFEND   $8          N  EFFCTV END DATE
 UPDDATE  $10         N  UPDATE DATE
 UPDTIME  $8          N  UPDATE TIME
 OP_ID    $8          N  UPDATE OPER ID
 BPS_KYWD $5          N  BPS KEYWORD
 INC_VAL  $1          N  INCLUDE VALUE
 EXC_VAL  $1          N  EXCLUDE VALUE
 CNFDTLAC $1          N  CONFIDENTIAL FLAG
 REG_ACC  $1          N  REGULATED FLAG
 PASSWORD $8          N  PASSWORD
 PWDATE   $8          N  PASSWORD UPD DATE
 FUN_CODE $3          N  FUNCTION CODE
```

Logging users

We want to keep track of who is utilizing our system and what they are doing. As users logon and navigate throughout the INTRANET, each program writes one line to the WEBLOG dataset.

```
%MACRO CHECK;
  %IF &SQLOBS EQ 0 %THEN %DO;
     DATA ONE;
       LENGTH USERID $ 8 LINK $200;
       USERID = UPCASE(SYMGET('USERID'));
       LOGDATE = DATE();
       LOGTIME = TIME();
       LINK = 'UNSUCCESSFUL LOGIN ATTEMPT';
     RUN;

     PROC APPEND BASE=PRIVATE.WEBLOG DATA=ONE;
     RUN;
    %END;
 /* write error message to browser */
%MEND CHECK;
```

The WEBLOG dataset is fairly simple, it merely logs userid, current date and time, and page visited. Each program in our Intranet writes one record to this dataset. The LOGON.SAS program will register when a user logs on and whether or not the attempt was successful. We can run reports off of this file to see what kinds of links are being followed and who is following them and judge how much a particular function is being utilized. We have already used this dataset to isolate hackers who were trying to get in after hours!

Writing a "cookie" dataset

How do we pass userid from page to page in order to log it correctly? We looked into writing a cookie to the user's hard drive and reading it into each page. However, users can turn off cookies in their browsers and we had little or no control over other information in the cookie. We could pass it using hidden fields but had problems when users used the back button on the browser. After much debate, we finally settled on writing a "cookie" dataset into our temporary directory on the Alpha. The dataset name is built by reversing the IP address of the user's PC (provided in the macro variable _RMTADDR), concatenating a letter , and truncating it to 8 characters. This dataset has one observation and can be updated by any SAS program running through the broker.

LOGON.SAS Excerpt 2

```
  /* build work dataset name */
DATA _NULL_;
  LENGTH W1 W2 $ 12 DSN $ 8;
  W1 = COMPRESS(SYMGET('_RMTADDR'),'.');
  W2 = LEFT(TRIM(REVERSE(W1)));
  DSN = 'K' || SUBSTR(W2,1,7);
  CALL SYMPUT('TEMPDSN',DSN);
RUN;

 /* BUILD PERMANENT WORK DATASET */
DATA SASTEMP.&TEMPDSN(KEEP=USERID
                           EXPDATE);
    ATTRIB USERID LENGTH=$8 LABEL='USER ID'
           EXPDATE LENGTH=$8
                   LABEL='EXPIRATION DATE';
    USERID = SYMGET('USERID');
    TODAY = DATE();
    EXPDATE =
        COMPRESS(PUT(TODAY,YYMMDD10.),'-');
RUN;
```

Each time the user opens a new browser session and logs into the INTRANET, the "cookie" dataset will get reinitialized. We did this because we want to discourage multiple sessions on the same PC and because there is really no other way to distinguish browser sessions on the same machine.

The _RMTADDR macro variable is available to every program. Thus, the "cookie" dataset and the information contained in it can be accessed by any program that goes through the broker. Other programs within our Intranet also write information to the dataset.

Dynamically building the menu screen

Once a user has made it past all the checks, a dynamic menu screen is built with options for each link that user has access to. Recall the FUN_CODE field in the APPLFUNC dataset. There is one record in the APPLFUNC dataset for each code that user has access to. For example, people who only look at the daily statistical reports have access function codes ID1, SD1, CD1, and ALL.

The DYNamic HTML dataset which also lives on the Alpha, contains all links available to our INTRANET

```
          APPLFUNC DATASET EXCERPT

  USERID  PASSWORD APP_CODE  FUN_CODE

  POOKIE     TUNA   SASWEB      ALL
  POOKIE     TUNA   SASWEB      CD1
  POOKIE     TUNA   SASWEB      ID1
  POOKIE     TUNA   SASWEB      SD1
```

```
Variable Length     Key Label

LINK     $200        N  Hypertext link
TEXT     $200        N  HTML Text to print
FUN_CODE $3          N  Function code
ORDER      8         N  Print order on pg
GROUP      8         N  Group code
```

users. Each link has an associated function code. Users with a matching function code will get those links displayed on their menu.

The order and group variables are used to group links under a specific heading and order them accordingly.

The following BUILDPG macro shows how the APPLFUNC and DYNHTML datasets are merged together to pull all links for this particular user and then dynamically build the main menu page. This allowed us to customize the menu for each user without programming for each user.

```
%MACRO BUILDPG;

 /* get all access codes for this user */
DATA ONE;
  SET PRIVATE.APPLFUNC(READ=WEBS);
  IF USERID EQ UPCASE(SYMGET('USERID'));
RUN;

PROC SORT DATA=ONE;
  BY FUN_CODE;
RUN;

PROC SORT DATA=PRIVATE.DYNHTML OUT=DYN;
  BY FUN_CODE;
RUN;

 /* merge user with all available links
    matching by access codes */
DATA TWO(DROP=APP_CODE EFFBEGIN EFFEND
           UPDDATE UPDTIME OP_ID BPS_KYWD
           CNFDTLAC REG_ACC PASSWORD);
  MERGE ONE(IN=A)
        DYN(IN=B);
  BY FUN_CODE;

  IF A AND B;

/* SO IF THERE ARE CODES DEFINED BUT NOT
   YET IN SERVICE, THIS WILL
   THROW THEM OUT */
  TODAY = DATE();
  IF EFFEND EQ ' ' THEN
    EFFEND =
      COMPRESS(PUT(TODAY,YYMMDD10.),'-');
 /* DEFAULT EFFECTIVE END DATE */

  IF INPUT(EFFBEGIN,YYMMDD8.) LE TODAY
    LE INPUT(EFFEND,YYMMDD8.);
             /* MAKES SURE THIS CODE */
             /* IS STILL VALID */
RUN;

%NUMOBS(TWO);

 /* IF ALL CODES HAVE EXPIRED THEN THIS
USER IS NO LONGER VALID */

%IF &NUM EQ 0 %THEN %DO;
  /* LOG IT TO WEBLOG */
  /* SEND A PAGE TO THE BROWSER THAT */
  /* LOGIN FAILED */
     .
     .

%END;

PROC SORT DATA=TWO;
  BY GROUP ORDER FUN_CODE TEXT;
RUN;

DATA _NULL_;
  SET TWO END=EOF;
  BY GROUP ORDER FUN_CODE;

  /* DYNAMICALLY BUILD PAGE */
  FILE _WEBOUT;
```

```
  IF _N_ =1 THEN DO;
    PUT 'Content-type: text/html';
    PUT;
    PUT '<HEAD>';
    PUT '<TITLE> LabOne LIVE!</TITLE>';
    PUT '<LINK REL=STYLESHEET'
        ' TYPE="text/css" HREF='
    '"http://isnt/sas/sasstandard.css">';
    PUT '<STYLE>';
    PUT '.USER {vertical-align: top; ';
    PUT '       font-weight: bold; ';
    PUT '       font-family: arial; ';
    PUT '       }';
    PUT '</STYLE>';
    PUT '<H1>';
    PUT '<IMG border="0" SRC='
  '"HTTP://172.24.1.14/WWW/LABONE.GIF">';
    PUT '<img border="0" src='
              'http://isnt/sas/live.gif'
              'width="301" height="80">';
    PUT '<BR> ';
    PUT '<SPAN>L</SPAN>aboratory';
    PUT '<SPAN>I</SPAN>nformation';
    PUT '<SPAN>V</SPAN>isualization';
    PUT '<SPAN>E</SPAN>nvironment';
    PUT '</H1>';
    PUT '</HEAD>';
    PUT '<HR CLASS="HR">';
    PUT '<BODY>';
     /* USERID BANNER LINE */
    PUT '<P CLASS="USER">'
       'Lab<I>One</I> LIVE! user: '
       '<FONT SIZE=-1 COLOR="MAROON"><U>'
       "%upcase(&USERID)"
       '</U></FONT></P><BR>';
    PUT '<CENTER><TABLE CELLPADDING="5"'
        ' CELLSPACING="10">';
  END;
  IF ORDER = 1 THEN DO;
   /* THIS IS A HEADER */
    PUT '<TR><TH VALIGN="TOP">'
        '<EM><STRONG><BR>'
        '<FONT SIZE=3 COLOR="BLACK">'
        LINK '</FONT></A>'
        '</STRONG></EM></TH></TR>';
  END;
  ELSE
    PUT '<TR>'
       '<TD VALIGN="TOP" ALIGN="CENTER">'
       '<A HREF="' LINK '">'
       '<FONT COLOR="MEDIUMBLUE">' TEXT
       '</FONT></A></TD></TR>';
  IF EOF THEN DO;
    PUT '</TABLE></CENTER>';
    PUT '</BODY>';
    PUT '</HTML>';
  END;
RUN;
%MEND BUILDPG;
```

Menu screen

Since the menu screen is generated by the broker, it cannot be bookmarked and our security cannot be bypassed. We are still working on broker generated links from the menu itself which cannot be bookmarked. By writing the selected link to the "cookie" dataset, we could generate the next HTML page from the broker, thus preventing the bookmarks.

Conclusion

This paper has shown an extremely simple way to implement Intranet security without using encryption. Since our LabOne LIVE! System is an Intranet, we have some amount of control over who will have access. This allows us to get by without implementing encryption. For sensitive data passed over the Internet, encryption is definitely a necessity. Encryption is currently not available at our site. Some day it might be and at that time we would like to utilize it.

Acknowledgements

SAS, SAS/Intrnet, and SAS/SHARE are registered trademarks or trademarks of SAS Institute, Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

References

SAS® Institute Web Tools Documentation

SAS Institute Inc., *SAS® Guide to Macro Processing, Version 6, Second Edition,* Cary, NC: SAS Institute Inc., 1990. 319pp.

SAS Institute Inc., *Running SAS® Applications on the Web Course Notes,* Cary, NC: SAS Institute Inc., 1997.

Laura Lemay, *Web Publishing with HTML 3.0, Second Edition,* Indianapolis, IN: Sams.net Publishing, 1996.

Dr. Joe Burns, *http://www.htmlgoodies.com*

Author

Kerril Bauerly
LabOne, Inc.
10101 Renner Blvd
Lenexa, KS 66219
Phone: (913) 888-1770 ext. 1318
E-mail: kerril.bauerly@labone.com