Paper 195

# Intranet Security and SAS®: A Sneaky Approach to Dynamic Applications

**Kerril Bauerly, LabOne, Inc.,Lenexa, KS**

### *Introduction*

When we first started developing our LabOne, LIVE! Intranet, security was our biggest concern. As we became more familiar with the tools available to us through SAS/IntrNet, we developed new ways of making our system secure. Our login application is covered in another paper, but this paper covers what happens after a user has successfully logged into our Intranet.

This Intranet system uses the SAS broker and application dispatcher available with SAS/IntrNet software in addition to the HTML formatting macros. Due to the sensitivity of our data, security is a big issue. We want to offer users an easy to use interface and still keep unauthorized users from viewing sensitive materials.

It's important to note here that our datasets are stored on an Open_VMS Alpha box. These datasets can then be accessed from the broker through the SAS/SHARE server. In addition, these datasets are password protected for both read and write. We are regulating user access with 2 main datasets: our APPLication FUNCtion dataset and our DYNamic HTML dataset.

### *APPLFUNC dataset*

The APPLFUNC dataset contains one observation for each user for each function that the user is allowed to perform. Functions are defined by the FUN_CODE field in the dataset. When a new user logs on, all observations belonging to that userid are subsetted from APPLFUNC and used to build a main menu unique to that user.

**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 |

### *DYNHTML dataset*

The DYNHTML dataset contains one observation for each possible line in the main INTRANET menu. These lines consist of headers, text, and links to other pages. Links are grouped using the GROUP code and print in the order specified in the ORDER variable. There is one heading for each group, which has an order value of 1 so it will print first. The FUN_CODE field in this dataset corresponds to the FUN_CODE field in the APPLFUNC dataset. FUN_CODE determines access while GROUP and ORDER are variables used to make the page display nicely.

**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 |

**Libref: DATATEST**
**Dataset: DYNHTML**

| 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 |

An individual user's APPLFUNC observations are merged with the DYNHTML dataset by FUN_CODE and passed to the code that builds the main menu. GROUP and ORDER, which allows us to display items in the order we want, sort the menu items. A DATA _NULL_ then builds our web page and our users receive only the links they are allowed to visit.

The best part about all of this is that the menu page cannot be bookmarked. The user must login in order to get to the main menu because the SAS program running from the broker must build it.

***LOGON.SAS program***

The LOGON.SAS program combines the APPLFUNC and DYNHTML datasets and builds the main menu dynamically. The details of this program are covered in the previous paper. However, it is important to mention here that this program uses a "cookie" dataset which lives in a 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 read or updated by any SAS program running through the broker.

***Static pages on the Alpha***

Since our goal is to keep users from bookmarking pages behind the logon screen, we try to build every page as we go. This means every link that is clicked issues a call to the SAS broker. While it can sometimes be time-consuming, we feel that the payoff in security is worth the few extra seconds that it may take to load the new page.

Several of our web reports are generated as part of nightly production. These are programs that would be too time-consuming to run through the broker or need data only available in Oracle. The output is generated using the SAS web macros and stored in HTML form on the Alpha. The links to the Alpha pages are stored as calls to the broker in DYNHTML. The links call a macro, which performs the necessary FTP transfer when the page needs to be loaded.

Because we cannot completely prevent users from bookmarking the pages, we verify that the user trying to download the page does indeed have the authority to see it. This code is similar to the code in LOGON.SAS. The access code (FUN_CODE) associated with the link is passed in, the USERID is obtained from the "cookie" dataset, and the 2 are compared with the APPLFUNC dataset to verify user access.

```
DYNHTML.DATA 1

LINK:     http://isnt/cgi-bin/broker.exe?
_service=test&_program=test.ftpdown.sas&
_debug=0&indsn=insmenu.htm&
indir=sas_webroot&fun_code=ID1

TEXT:     Daily Insurance Statistical
Reports

FUN_CODE: ID1
ORDER:               2
GROUP:              10
```

The code to FTPDOWN.SAS is located in Appendix I. A sample observation from the DYNHTML dataset which calls FTPDOWN is shown. The value of the LINK variable contains several parts. The first is the call to the broker:

**`http://isnt/cgi-bin/broker.exe`**

The ? separates the broker call from the parameters that are passed into the broker program. The first parameter is **`_service=test`**. This is a SAS system variable and tells the broker which application server to send the call to. We have both test and production environments set up at our site, so **`_service=test`** routes this call to the application server in our test environment. **`_service=prod`** would route this call to our production environment.

Each subsequent parameter in the list is separated by ampersands (&). The next parameter is **`_program=test.ftpdown.sas.`** **`_program`** is also a SAS variable needed by the broker. It tells the application server which program to execute. In this case **`test.`** is the libref assigned in our SRVAUTO.SAS program. Again, this points to a test directory. The name of the program to execute is **`ftpdown.sas`**. **`_debug = 0`** is another SAS variable useful for debugging. Possible values for **`_debug`** are listed with the SAS/IntrNet documentation. We use a value of 131 for testing and 0 once a program has been moved to production.

The remaining parameters are needed by the SAS program FTPDOWN. INDSN is the name of the file to be FTP'd. INDIR is the directory where the file resides on the Alpha. FUN_CODE is described below.

***A few notes about the code***

This program first performs a check to see if the user is authorized to view the requested page. It does this by pulling USERID from the "cookie" dataset and the FUN_CODE value that was passed in through the broker call. These 2 values are compared to the same values in the APPLFUNC dataset through the PROC SQL. If there is not a match (&SQLOBS=0), this particular user is not authorized to view the requested page. A note is logged in the WEBLOG dataset and a message is written to the browser informing the user that he is in trouble!

If the user does indeed have the authority to view the page, an FTP between the Alpha box and the browser is initiated and the page is displayed. In the FILENAME statement, the device type is specified as FTP and all necessary parms are specified.

```
FILENAME CP ftp "&INDSN" debug
         /* IP ADDRESS OF THE ALPHA */
         host='999.99.9.99'
         /* USERID AND PASSWORD, WE
           HAVE A SPECIAL ONE SETUP
           JUST FOR FTP */
         user='XXXXX' pass='XXXXX'
          /* DIRECTORY TO READ ON THE
             ALPHA */
         cd="&INDIR";
```

The "cookie" datasets are cleaned up nightly. A user can bookmark a page from the Alpha one day. If he tries to access it the next day without first logging in, the program will fail because it cannot find the "cookie" dataset with the userid.

***Static pages on the Server***

A limited number of static pages exist on the server side. These are handled the same way as pages from the Alpha.

```
DYNHTML.DATA 2

LINK:     http://isnt/cgi-bin/broker.exe?
_service=test&_program=test.ftpacross.sas
&_debug=131&indsn=cntrlcht.htm&
indir=test&fun_code=ALL

TEXT:     Laboratory Instrument Control
Charts

FUN_CODE: ALL
ORDER:               2
GROUP:              35
```

The program to download these files is called FTPACROSS.SAS. A listing of this program is in Appendix II. A sample observation from the DYNHTML dataset which calls FTPACROSS is shown above.

The first three parameters in this LINK value are the same as for FTPDOWN. The only difference is in the **_program=** parameter which points to the FTPACROSS program. The last 3 parameters are unique to the FTPACROSS program. Again INDSN is the file to write to the browser and FUN_CODE is the access code associated with this link. INDIR is the directory on the server where the HTML file to be copied resides.

***A few notes about the code***

This program performs the exact same checks as FTPDOWN to see if the user is authorized to view the requested page. The only significant difference between these programs is the FILENAME statement.

The device type is specified as URL instead of FTP and the port number of our proxy server is specified immediately following the server name (:80). This allows us to read the requested file from the HTTPD server.

```
FILENAME CP URL
"http://isnt:80/sas/&indir/&indsn" DEBUG;
```

***Maintenance of DYNHTML and APPLFUNC datasets***

Because our system is dynamic and changes daily, we have developed several maintenance programs which allow us to add, update, and delete links and add and update users through the browser. However, these are beyond the scope of this paper.

***Conclusion***

While this method may not be the best or the most elegant, it does work quite well for our purposes. We are able to regulate who has access to our INTRANET and what they have access to. In addition, we have tried to make it as difficult as possible for users to set a bookmark behind the logon screen.

**Acknowledgements**



**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

## Appendix I - FTPDOWN.SAS

```
/*******************************************************************************/
/* LabOne LIVE!                                                                */
/*******************************************************************************/
/* FTPDOWN.SAS - STANDARD MACRO TO FTP A FILE FROM THE ALPHA BOX DOWN TO THE NETWORK   */
/*               THIS ALLOWS US TO WRITE STATIC HTML PAGES ON THE ALPHA BOX, BUT COPY  */
/*               THEM DIRECTLY TO THE BROWSER TO HIDE THE SOURCE                       */
/*                   PARMS: DSN = FILENAME ON THE SAS ALPHA BOX                        */
/*                          SASDIR = DIRECTORY ON THE SAS ALPHA BOX                    */
/*                   HOST, USERID, AND PASSWORD STAY THE SAME UNLESS BRIAN CHANGES THEM*/
/*******************************************************************************/

OPTIONS COMAMID=TCP MPRINT;

 /* INITIALIZE DIRECTORIES ON THE ALPHA BOX (NAMED SAS) TO THE SHARE SERVER */
LIBNAME SASTEMP 'SAS_TEMP:' SERVER=SAS.SAS_RMT;
LIBNAME DATA 'SAS_DATA1:[SAS_DATA.TEST]' SERVER=SAS.SAS_RMT;

%GLOBAL INDSN INDIR FUN_CODE USERID;

%LET INDSN=%SUPERQ(INDSN);
%LET INDIR=%SUPERQ(INDIR);
%LET FUN_CODE=%SUPERQ(FUN_CODE);

 /* BRING IN CODE TO BUILD THE "COOKIE" DATASET NAME */
%INCLUDE 'D:\SAS\INTRNET\PROD\BLDTNAME.SAS';

DATA _NULL_;
  SET SASTEMP.&TEMPDSN;
  CALL SYMPUT('USERID',USERID);
RUN;

  /* COMPARE USERID FROM "COOKIE" DATASET AND FUN_CODE PASSED IN WITH SECURITY DATASET */
PROC SQL NOPRINT;
  SELECT USERID, FUN_CODE
  FROM DATA.APPLFUNC(READ=WEBS)
  WHERE UPCASE(USERID) = UPCASE(SYMGET('USERID')) AND
        UPCASE(FUN_CODE) = UPCASE(SYMGET('FUN_CODE')) AND
        APP_CODE = 'SASWEB';
QUIT;

%MACRO CHECK;
   /* IF NO OBS, NO MATCH, INVALID USER */
  %IF &SQLOBS EQ 0 %THEN %DO;
     DATA ONE;
       LENGTH USERID $ 8 LINK $200;
       USERID = UPCASE(SYMGET('USERID'));
       LOGDATE = DATE();
       LOGTIME = TIME();
       LINK = "INVALID ACCESS ATTEMPT TO &INDSN";
     RUN;

      /* LOG TO TRACKING DATASET */
     PROC APPEND BASE=DATA.WEBLOG DATA=ONE;
     RUN;

       /* DISPLAY NASTY MESSAGE */
     DATA _NULL_;
       FILE _WEBOUT;
       PUT 'Content-type: text/html';
       PUT;
       PUT
          '<LINK REL=STYLESHEET TYPE="text/css" HREF="http://isnt/sas/sasstandard.css">';
       PUT '<BODY BACKGROUND="http://isnt/sas/sasback_teal3.gif">';
       PUT '<H1><FONT COLOR=RED> Breach of Security Attempt </FONT></H1>';
       PUT '<P>';
       PUT 'You are not authorized to view this page.  Security violation has been'
           ' noted. </P>';
       PUT 'Use the back button on your browser';
       PUT 'to return to the previous page';
       PUT '</BODY>';
```

```
      PUT '</HTML>';
     RUN;
   %END;
   %ELSE %DO;

FILENAME CP ftp "&INDSN" debug host='999.99.9.99'
                user='XXXXX' pass='XXXXX'
                cd="&INDIR";

DATA _NULL_;
  /* READ CANNED FILE IN, WRITE IT RIGHT TO THE BROWSER */
  FILE _WEBOUT;

  PUT 'Content-type: text/html';
  PUT;

  INFILE CP LENGTH=LEN END=EOF;
  DO UNTIL (EOF);
    INPUT HTMLINE $VARYING200. LEN;
    LEN = LENGTH(HTMLINE);
    PUT HTMLINE $VARYING200. LEN;
  END;
  STOP;
RUN;

%END;
%MEND CHECK;

%CHECK
```

## Appendix II - FTPACROSS.SAS

```
/*******************************************************************************************/
/* LabOne LIVE!                                                                            */
/*******************************************************************************************/
/* FTPACROSS.SAS - STANDARD MACRO TO COPY A FILE FROM THE NETWORK                          */
/*                 DIRECTLY TO THE BROWSER TO HIDE THE SOURCE                              */
/*                     PARMS: INDSN = FILENAME                                             */
/*                            INDIR = NETWORK DIRECTORY                                    */
/*******************************************************************************************/

OPTIONS COMAMID=TCP MPRINT;

 /* INITIALIZE DIRECTORIES ON THE ALPHA BOX (NAMED SAS) TO THE SHARE SERVER */
LIBNAME SASTEMP 'SAS_TEMP:' SERVER=SAS.SAS_RMT;
LIBNAME DATA 'SAS_DATA1:[SAS_DATA.TEST]' SERVER=SAS.SAS_RMT;

%GLOBAL INDSN INDIR FUN_CODE USERID;

%LET INDSN=%SUPERQ(INDSN);
%LET INDIR=%SUPERQ(INDIR);
%LET FUN_CODE=%SUPERQ(FUN_CODE);

 /* BRING IN CODE TO BUILD THE "COOKIE" DATASET NAME */
%INCLUDE 'D:\SAS\INTRNET\SAMPLE\BLDTNAME.SAS';

DATA _NULL_;
  SET SASTEMP.&TEMPDSN;
  CALL SYMPUT('USERID',USERID);
RUN;

  /* COMPARE USERID FROM "COOKIE" DATASET AND FUN_CODE PASSED IN WITH SECURITY DATASET */
PROC SQL NOPRINT;
  SELECT USERID, FUN_CODE
  FROM DATA.APPLFUNC(READ=WEBS)
  WHERE UPCASE(USERID) = UPCASE(SYMGET('USERID')) AND
        UPCASE(FUN_CODE) = UPCASE(SYMGET('FUN_CODE')) AND
        APP_CODE = 'SASWEB';
QUIT;
```

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

      /* LOG TO TRACKING DATASET */
     PROC APPEND BASE=DATA.WEBLOG DATA=ONE;
     RUN;

       /* DISPLAY NASTY MESSAGE */
     DATA _NULL_;
       FILE _WEBOUT;
       PUT 'Content-type: text/html';
       PUT;
       PUT
         '<LINK REL=STYLESHEET TYPE="text/css" HREF="http://isnt/sas/sasstandard.css">';
       PUT '<BODY BACKGROUND="http://isnt/sas/sasback_teal3.gif">';
       PUT '<H1><FONT COLOR=RED> Breach of Security Attempt </FONT></H1>';
       PUT '<P>';
       PUT 'You are not authorized to view this page.  Security violation has been '
           'noted. </P>';
       PUT 'Use the back button on your browser';
       PUT 'to return to the previous page';
       PUT '</BODY>';
       PUT '</HTML>';
     RUN;
   %END;
   %ELSE %DO;

FILENAME CP URL "http://isnt:80/sas/&indir/&indsn" DEBUG;

DATA _NULL_;
  /* READ CANNED FILE IN, WRITE IT RIGHT TO THE BROWSER */
  FILE _WEBOUT;

  PUT 'Content-type: text/html';
  PUT;

  INFILE CP LENGTH=LEN END=EOF;
  DO UNTIL (EOF);
    INPUT HTMLINE $VARYING200. LEN;

    LEN = LENGTH(HTMLINE);
    PUT HTMLINE $VARYING200. LEN;
  END;
  STOP;
RUN;

%END;
%MEND CHECK;

%CHECK
```