FTP Security and Unix Login Control

Great Lakes User’s Group

Don Prezioso

Baldwin-Wallace College
275 Eastland Road
Berea, OH 44017
440-826-2310

E-mail: dprez@bw.edu
**Table of Contents**

<table>
<thead>
<tr>
<th>Introduction</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disclaimer</td>
<td>1</td>
</tr>
</tbody>
</table>

**FTP Security / Login Control**

<table>
<thead>
<tr>
<th>/etc/profile</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>datatel.login</td>
<td>3</td>
</tr>
<tr>
<td>bwc_udt.c</td>
<td>4</td>
</tr>
<tr>
<td>LOGIN.EXECUTE</td>
<td>7</td>
</tr>
<tr>
<td>set.access</td>
<td>8</td>
</tr>
<tr>
<td>&amp;MYHOME&amp;</td>
<td>9</td>
</tr>
<tr>
<td>lock.logins</td>
<td>10</td>
</tr>
<tr>
<td>unlock.logins</td>
<td>11</td>
</tr>
</tbody>
</table>
Introduction

Over the last 14 years I have spent a great deal of time trying to make the systems I run more secure without making things any harder on the users than they have to be. Having changed operating systems, database systems, R13-R16, and the college I work for all in the last 5 years, I have needed to re-create a number of the tools I have developed to ensure that the users of the system can do their jobs yet also provide for a secure system. This presentation is more or less a collection of the programs and scripts that I have found most valuable in getting that job done.

These examples show how we have been able to give our users access to FTP and other methods for downloading and uploading files with our Colleague machine without giving them a dangerous level of access to our database files.

Disclaimer

While the examples given here have been tested and work properly on the Colleague system at Baldwin-Wallace College, we cannot guarantee that they will work properly on any other system. Differences in operating system, release level, security setup, and account structure in particular may cause these examples to work improperly.
FTP Security / Login Control

When users are utilizing Colleague, they need full (read and write) access to all (or at least most) of the database files. As long as they remain in UniData, there are protections that can be put in place to keep users from deleting or damaging your institution’s data files. The problem arises when those users need access to the system when they are NOT in UniData. The example we are using in this case is FTP, but it could easily be any time a user needs access to the system when they are not in the UniData environment.

To solve this problem, we use a combination of Unix scripts, a C program, and a UniBasic program to give users access to the Colleague database files ONLY while they are in UniData. When they are NOT in UniData (like during an FTP session) they will not have access to ANY of the database files with the exception of a few directories (like _HOLD_) that we want them to be able to access with FTP.

/etc/profile

To begin with, when users log in, they execute the system profile script. Of importance here is the section of the script that allows a limited group of people (generally I.T. staff) access to the operating system level. All other users then execute another script (datatel.login) using the exec command so that if they should somehow abort from that script, or from UniData, they will be without a shell, and thus be logged out. Note that using the system profile script means that the user never executes their personal profile script from their home directory since they could modify that. Also note that we place /datatel/bin in the PATH environment variable for all users BEFORE the path to the UniData executables. That way we can put our own version of the ‘udt’ command there and it will be executed instead of the real thing.

```
PATH=/usr/local/bin:/datatel/bin:/usr/ud41/bin: { rest of path removed } . /usr/udms/udmsprofile case $USER in
  root | datatel | adm | udsmsmgr ) ;;
  katchins | ckitko | kmagpoc | scoyne ) ;;
  dppez | mshadrak | ragnew | dmushat | dkilbane ) ;;
  jszloh | qjaamey | lartim | jknaur | srobinso ) ;;
  allofus | Reggie )
    exec /datatel/bin/datatel.login ;;
  *) stty > /dev/null
    if [ 0 -eq $? ] ; then
      exec /datatel/bin/datatel.login
    else
      cd /datatel/live/collive
    fi
  fi
esac
stty intr "^C"
stty quit "^\n"
stty susp "^Z"
if [ 0 -ne $? ] ; then
  cd /datatel/live/collive
fi
```
datatel.login

Most users will continue on to the datatel.login script. The first thing this script does is to check if the user in question is one of those allowed to access multiple accounts (like coldev or coltest). If they are not one of the users listed in the file menu.users (a text file with one username per line) they are immediately taken to the live main account where UniData is started. This is not the real ‘udt’ command remember, this is our own version. If they are allowed to access multiple accounts, then they are given a menu to choose from, which determines where they are taken, and again, our version of UniData is started.

```bash
#!/bin/ksh
TERM=bwvt420 ; export TERM
if [[ -z $( grep ^$USER /datatel/menu.users ) ]] ; then
cd collive ; exec udt ; exit
fi
while true; do
  clear
  cat <<end_of_menu
You are seeing this menu because you have access to several accounts. If you only wish to access the LIVE account, and do not want to see this menu anymore, send an E-mail message to dprez@bw.edu and I will remove this menu for you.

Which Datatel Account would you like to start?"

L) LIVE Account"
T) Test Account"
D) Development Account"
C) Enrollment Chill Account"
P) Change Password"
X) Exit"

end_of_menu

print " Enter L, T, D, C, P, or X: \c"
read ANS
case $ANS in
  E | e ) cd coleduc16 ; exec udt ; exit ;;
  T | t ) cd coltest ; exec udt ; exit ;;
  L | l ) cd collive ; exec udt ; exit ;;
  D | d ) cd coldev ; exec udt ; exit ;;
  C | c ) cd chill ; exec udt ; exit ;;
  P | p ) passwd ;;
  X | x ) exit ;;
  *) print "Invalid: Try again!" ;;
esac
sleep 2
done
exit
```
bwc_udt.c

This is our ‘replacement’ for the standard udt command. We create a link to the compiled program from /datatel/bin/udt, and since /datatel/bin is in the search path before the standard udt command this is what gets run. This program gives the user the groups needed to access the database files, then starts the ‘real’ udt command.

Note that this program uses some AIX specific functions (like ‘getgroups’ and ‘setgroups’) that may be different on other operating system. For example, Linux also has getgroups and setgroups functions that do the same thing, but some of the constants may be different. Also, some older flavors of unix do not allow a user to belong to multiple groups at the same time. Because of this, if you are not using AIX, you will need to make some changes to the program, but the same concepts should apply.

/*
Written by Don Prezioso for Baldwin-Wallace College, Jan 4, 1999.
This program gives a user membership in some groups that are needed for the user to access Datatel's database programs and files, and enters the UniData environment. The program also checks for the existence of two files, /datatel/login.txt, and /datatel/userlock.txt. If /datatel/login.txt exists, then it is printed out and the program waits for input to continue. If /datatel/userlock.txt exists, it is printed and the program waits 5 seconds, then logs the user out.

To decide which groups a user should belong to the program checks the group of the VOC file in the current directory. The groups given to the user are this one, and the group of the file $UDTBIN/udt. If either file cannot be found, then the program exits.

To use this program, it should be called after changing directory to the account the user will be using. For example:

cd /datatel/live/collive
exec /datatel/bin/bwc_udt

Running the program this way causes the current shell to be replaced by this program. That way, when the program exits (for any reason) the user will be logged out.

To re-compile the program do the following as root

    xlc -Oo /datatel/bin/bwc_udt /datatel/bin/bwc_udt.c
    chown root:security /datatel/bin/bwc_udt
    chmod u+s /datatel/bin/bwc_udt
*/

#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <grp.h>
#include <unistd.h>
#include <limits.h>
#define LOCK_FILE_NAME "/datatel/userlock.txt"
#define MESSAGE_FILE_NAME "/datatel/login.txt"
#define MAX_LINE_LEN 255
#define GROUP_ERROR UID_MAX

int add_file_group ( const char *file_path )
/*
 * Get the group id of the file named by 'file_path' and add it to
 * to the groups for this process.
 */
{
    int status = 0;
    int idx = 1;
    int gid_count;
    struct stat* file_info;
    struct stat fibuf;
    gid_t group_id;
    gid_t gid_list[NGROUPS_MAX];
    char temp_text[MAX_LINE_LEN];

    file_info = &fibuf;
    if ( stat( file_path, file_info ) )
    {
        sprintf( temp_text, "Error getting group for %s", file_path );
        perror ( temp_text );
        status = 1;
    }
    else
    {
        group_id = file_info -> st_gid;
        gid_count = getgroups( NGROUPS_MAX, gid_list );
        if ( gid_count == -1 )
        {
            perror( "Error getting group membership" );
            status = 1;
        }
        else
        {
            while ( ( idx < gid_count ) && ( gid_list[idx] != group_id ) ) idx++;
            if ( idx >= gid_count )
            {
                gid_list[gid_count] = group_id;
                if ( setgroups( ++gid_count, gid_list ) )
                {
                    perror( "Error setting group membership" );
                    status = 1;
                }
            }
        }
    }
    return( status );
}

int print_text_file( char *file_path )
/*
 * Check for the existence of the text file named in 'file_path'
 * and if it exists print it out to stdout. Also return true if
 */
file existed so appropriate action can be taken.
*/
{
int status = 0;
char text_file_line[MAX_LINE_LEN];
FILE *text_file;

text_file = fopen(file_path, "r");
if ( text_file )
{
    fgets ( text_file_line, MAX_LINE_LEN, text_file );
    while ( ! feof( text_file ) )
    {
        printf( "%s", text_file_line );
        fgets ( text_file_line, MAX_LINE_LEN, text_file );
    }
    fclose( text_file );
    status = 1;
}
return( status );
}

main ( int argc, char* argv[] )
{
char user_groups[MAX_LINE_LEN];
char temp_text[MAX_LINE_LEN];
char udt_path[MAX_LINE_LEN];
int errors = 0;

sprintf( udt_path, "%s/udt", getenv( "UDTBIN" ) );

if ( ! errors ) errors = ( print_text_file( LOCK_FILE_NAME ) );
if ( ! errors ) errors = add_file_group( udt_path );
if ( ! errors ) errors = add_file_group( "VOC" );
if ( ! errors )
{
    if ( print_text_file( MESSAGE_FILE_NAME ) )
    {
        printf( "\n\n%s", "Press RETURN to continue " );
        fgets ( temp_text, MAX_LINE_LEN, stdin );
    }
}
seteuid( getuid() );
argv[0] = "udt";
if ( ! errors ) execv( udt_path, argv );
if ( errors ) sleep( 5 );
exit;
LOGIN.EXECUTE

This is a program that was originally taken from the Colleague Administrator learning guide, which we have modified so that even if a user has no xxx.LOGIN VOC entry they will still run the ST application. This program is called as a part of the LOGIN paragraph in all of our accounts so that we can insure that users will be forced into a menu, and that they will therefore be validated against the UT.OPERS records.

```
* DEW
* This program is executed upon user login. It searches the VOC
* file for a paragraph named USERID.LOGIN, where USERID is the
* UNIX userid of the given user. If the paragraph exists it is
* executed. This code appears in the Handouts section of the
* Colleague Administrator student guide.
* David Waldron. April, 1996.
*
* Modified 10/25/99 by Don Prezioso to enter the ST menu if there
* is no xxx.LOGIN for the user. That way we only need xxx.LOGIN records
* for those users who either go straight to the colon prompt or to the
* CF menu.
*  
* OPEN ","VOC" TO F.VOC ELSE
*   ABORT "CANNOT OPEN VOC"
* END
*  
* VOC.KEY=@LOGNAME:.LOGIN"
*  
* READ PARA FROM F.VOC, VOC.KEY THEN
*   EXECUTE VOC.KEY
* END ELSE
*   EXECUTE "ST"
* END
*
END
```

Now, when we create new users at the operating system level, we DO NOT give them access to the database, or to UniData. The only way they get that access is by running the above programs. Since FTP access does not run the programs, then they do not have any access to the database files while using FTP. Also, if there is some other way that they might access our system, or if we wanted to allow them access to the operating system level, they would not be able to access the database.
set.access

At this point, users can only access files they create in directories they would normally be able to access. To allow them access to be able to download hold files, and upload files (like for Financial Aid imports) they will need some limited access to some specific directories. We open up access to the _HOLD_ file for reading only so users can download information, and we open access to a separate directory (FTP) completely for uploading.

Note that on the directories above those that are opened, we only give execute access to ‘others’ (chmod o+x directory). This means that they can ‘cd’ (or the ftp equivalent) through those directories, but they cannot see what files exist in them. That way if someone tries ‘cd /datatel/live/collive’ they can’t see anything, but if they ‘cd /datatel/live/collive/_HOLD_’ then they will be able to see the files in the hold directory. In other words, people trying to use FTP have to know where they are going; they are not able to just ‘snoop’ around.

#!/bin/ksh
#
# ASH:DVP Reset permissions on datatel directories.
#
echo "Changing mode for upper level directories"
chmod 775 /datatel
chmod 775 /datatel/work
chmod 775 /datatel/live

echo "Changing owner, group and mode for release directories"
chown -R datatel:clive /datatel/release
chmod -R 770 /datatel/release
chown -R datatel:cdev /datatel/development
chmod -R 770 /datatel/development

echo "Changing owner, group and mode for collive files"
chown -R datatel:clive /datatel/live/collive
chmod -R 770 /datatel/live/collive

echo "Changing mode for live _HOLD_ and FTP directories"
chmod o+x /datatel
find /datatel/live/* -type d -prune -exec chmod o+x {} \
find /datatel/live/*/._HOLD_ -type d -prune -exec chmod o+rx {} \
find /datatel/work/* -type d -prune -exec chmod g+s {} \
find /datatel/work/*/._HOLD_ -type d -prune -exec chmod g+s {} \
find /datatel/live/*/DATA -type d -prune -exec chmod o+x {} \
find /datatel/live/*/DATA/Data_S -type d -prune -exec chmod o+x {} \
find /datatel/live/*/DATA/Data_S/Sched25.Directory -exec chmod o+rx {} \

echo "Changing owner, group, and mode for coltest files"
chown -R datatel:cdev /datatel/work/coltest
chmod -R 770 /datatel/work/coltest
chmod o+x /datatel/work/coltest
find /datatel/work/* -type d -prune -exec chmod o+x {} \
find /datatel/work/*/._HOLD_ -type d -prune -exec chmod o+rx {} \
find /datatel/work/*/._HOLD_ -type d -prune -exec chmod g+s {} \

October 24, 2000
find /datatel/work/*/SAVEDLISTS -type d -prune -exec chmod g+s {} \;
find /datatel/work/*/FTP_ -exec chmod o+rwx {} \;
find /datatel/work/*/DATA -type d -prune -exec chmod o+x {} \;
find /datatel/work/*/DATA_DATA_S -type d -prune -exec chmod o+x {} \;
find /datatel/work/*/DATA_DATA_S/SCHED25.DIRECTORY -exec chmod o+rx {} \;

echo "Changing owner, group, and mode for coldev files"
chown -R datatel:cdev /datatel/work/coldev
chmod -R 770 /datatel/work/coldev
chmod o+x /datatel/work/coldev

echo "Changing owner, group, and mode for chill files"
chown -R datatel:cdev /datatel/work/chill
chmod -R 770 /datatel/work/chill
chmod o+x /datatel/work/chill

echo "Changing owner, group, and mode for education files"
chown -R datatel:ceduc /datatel/work/coleduc*
chmod -R 770 /datatel/work/coleduc*
chmod o+x /datatel/work/coleduc*
find /datatel/work/coleduc*/ -type d -prune -exec chmod o+x {} \;
find /datatel/work/coleduc/*/HOLD_- -type d -prune -exec chmod o+rx {} \;

&MYHOME&

Also, we have created a file pointer to the users home directory called &MYHOME& so most downloads can be done there, where they normally have access. Additionally, the user’s home directory is actually the home directory on our campus e-mail system (remotely mounted via NFS), which is also available to networked PCs (via SAMBA) as the user’s U:/ directory.
Now that we have a way to give people the access they need while in UniData, we will have times that we want to keep them out, for instance when loading patches, or new releases. One of the things built in to our ‘replacement’ udt command is a check for the existence of the file /datatel/userlock.txt. If this file exists, then the contents of that file are displayed for a little while, and then the user is logged out, without ever entering UniData. You could create that file by hand whenever you needed it, but this script will create a rather generic message more easily. This script will also stop the DMI listener so WebAdvisor users will also be blocked from accessing the system.

```
#!/bin/ksh
# lock.logins
#
# This script creates a file that is checked at login time
# to see if normal users can access the Administrative Database. The
# file, if it exists, contains a message telling them they are not allowed
# access.
#
lockfile=/datatel/userlock.txt
echo "" >> $lockfile
echo "" >> $lockfile
echo "" >> $lockfile
echo "" >> $lockfile
echo "" >> $lockfile
echo "" >> $lockfile
echo "" >> $lockfile
chgrp staff $lockfile
chmod 775 $lockfile

/datatel/bin/dmictl stop

# "Access for non-privileged users to UniData has been locked"
# "To allow users back in, use run /datatel/unlock.logins"
```

* * * NOTICE * * *
Sorry, but access to the Baldwin-Wallace Administrative Database is currently restricted.
For more information, please call Administrative Computing at 2310.
unlock.logins

Of course it would also be nice to have a script to get rid of that file and restart the listener...

```
#!/bin/ksh
# unlock.logins
#
# This script deletes a file created by the lock.logins script, allowing
# access to the Administrative Database.
#
rm /datatel/userlock.txt
/datatel/bin/dmictl start
echo "Users can now access the Administrative Database"
```

***

These scripts and programs are only one method for handling FTP security and login control. There are a number of other methods that are also effective including using more secure FTP server daemons, restricting what users can use FTP and not using FTP at all, but using some other form of file transfer instead (Kermit for example). Each method has its advantages and drawbacks that may make them more, or less suitable at your site. The best advice is to know your setup very well, and “Be afraid... Be VERY afraid”.

Finally, some pointers on where to go for information. The great thing about a user’s group like this is the fact that, if there is something you would like to do, someone may well have already done it. [ftp://ftp.cedarville.edu/pub/unidata](ftp://ftp.cedarville.edu/pub/unidata) has a great wealth of programs, subroutines etc. If you don’t find what you are looking for there, by all means, post a question on info-datatel. Even if you don’t find exactly what you are looking for, you may find a starting point. I doubt that many of these programs can be put to use at your site as-is. My real hope is that they start you thinking about the kinds of things that can be done so that you will go home and make more utilities that I can use!