Interested in learning more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Build a Web Interface to Allow Users to Change their Passwords (The Web Password Page)

The purpose of this paper is to show you (the System Administrator) how to break free from the mundane task of periodically changing user passwords (in keeping with good security practices from GIAC Security Essentials). This document is designed to show you step-by-step how to build a web page for users to update their passwords on a UNIX or Windows server, easily, securely and without spending too much money on software! You'll need to be a little resourceful gathering the mostly free and a little commercial software...

Copyright SANS Institute

Author Retains Full Rights

AD

Build a Web Interface to Allow Users to Change their Passwords
(The Web Password Page)

SANS Security Essentials GSEC Practical Assignment
Version 1.3 (amended December 12, 2001)
Updated to include inline content February 9th, 2009

Overview

The purpose of this paper is to show you (the System Administrator) how to break free from the mundane task of periodically changing user passwords (in keeping with good security practices from GIAC Security Essentials). This document is designed to show you step-by-step how to build a web page for users to update their passwords on a UNIX or Windows server, easily, securely and without spending too much money on software! You'll need to be a little resourceful gathering the mostly free and a little commercial software, understanding basic shell programming, and knowing a little about compiling/installing UNIX software.

Notice

This document serves as an example of what you can implement within your own place of work. Proprietary security configuration assumptions in this example may or may not be consistent with your own security configurations. I do not intend to show how your web server's security can be strengthened outside of the web server related components. There are many security configurations for the Apache httpd.conf file and the UNIX server it runs on in this paper. This paper only modifies the particular options/configurations intrinsic to setting up the Web Password Page. The paper only does so much!

Preliminary Notes

- This paper demonstrates how to build your own Web Password Page. Assembling it is kind of like building something with Lego® (http://shop.lego.com/default.asp). Although I didn't design each sub-component, I did design the overall construction - how it all fits together. It's nothing fancy and you could probably find someone who's already built something similar (I didn't). It's hard to design (or write about) anything someone hasn't already done before. I like it because past the initial appearance, it's totally my design. This isn't a new concept, just an easy one to construct yourself. I also like it because the group using it likes and uses it. It saves them time, it's cheap (inexpensive), and frees up their admins for more meaningful tasks.
- Apache's web server (http://httpd.apache.org) running on an SGI server (IRIX 6.5) will be used in all examples. Statistics gathered by Netcraft

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

(http://www.netcraft.com/survey) show Apache to be the most widely used web server (and has been since '96) on the Internet.

Totals for Top Servers Across All Domains*

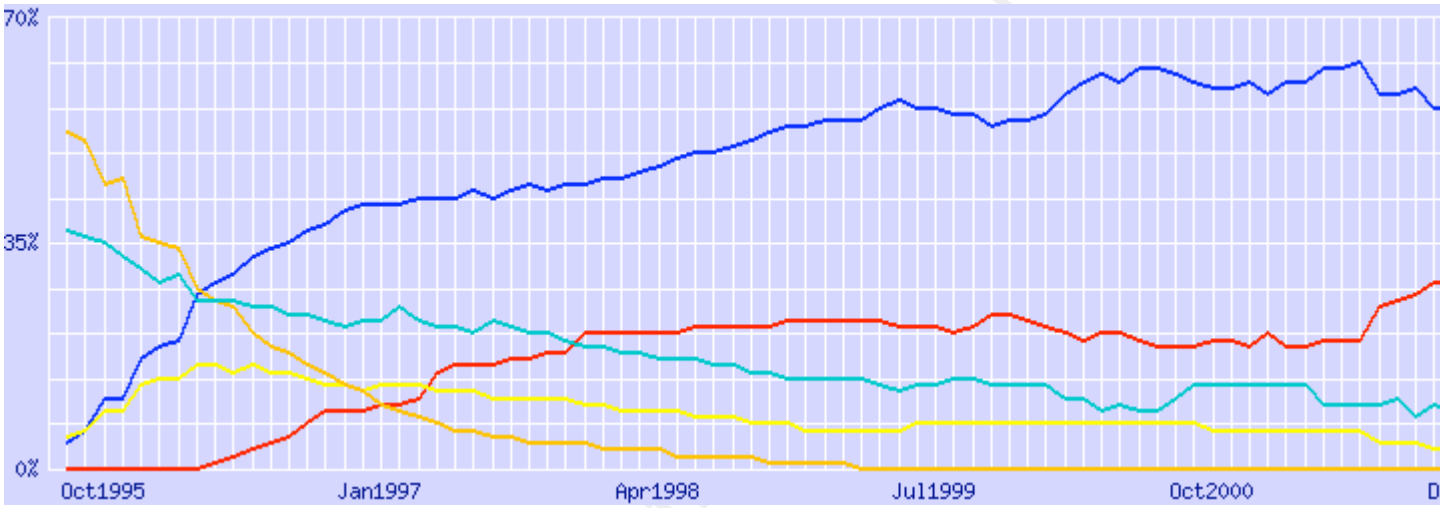

Figure 1:

"***iPlanet** is the sum of sites running iPlanet-Enterprise, Netscape-Enterprise, Netscape-FastTrack, Netscape-Commerce, Netscape-Communications, Netsite-Commerce & Netsite-Communications. **Microsoft** is the sum of sites running Microsoft-Internet-Information-Server, Microsoft-IIS, Microsoft-IIS-W, Microsoft-PWS-95, & Microsoft-PWS." [1]

[1] http://www.netcraft.com/survey

Table of Contents:

Section I. Build the Apache Web Server

Section II. Set Up CGI Scripts

Section III. Example Screen Shots

Section IV. Summary

Section V. Bibliography

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

Section I: Build the Apache Web Server

Overview

The Apache web server must be built using a couple of "add-ons": OpenSSL and mod_ssl. Here's a picture to help you "see" what you'll be building:

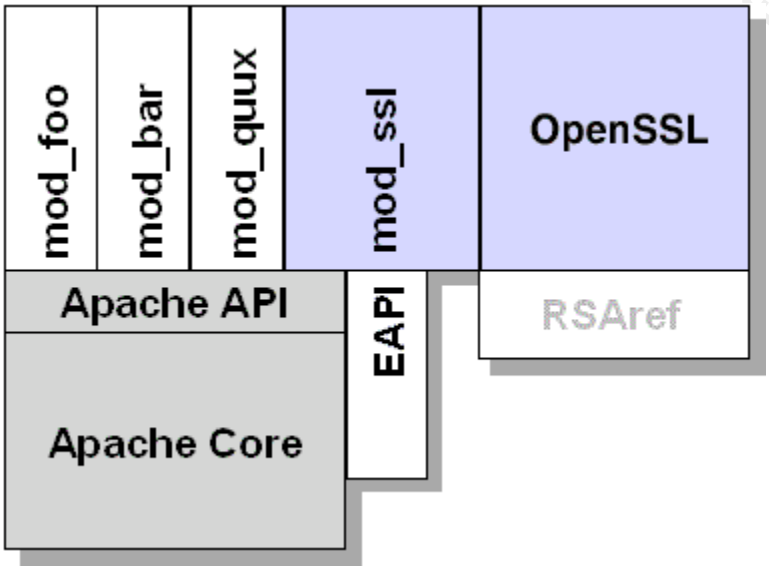

Figure 2: SSL Module Architecture Overview
(Altered image taken from http://httpd.apache.org/docs-2.0/ssl/ssl_overview.html)

Tasks

- Verify the digital signature of any signed files you get.
  - PGP Pretty Good Privacy implements encryption and authentication. GnuPG is a similar utility released under the GNU public licence. MD5 is a "Command-line utility that computes and checks *message digests* (digital signatures) generated by the MD5 algorithm as defined by RFC 1321." - John Walker, founder of Autodesk, Inc. (http://www.fourmilab.ch/autofile) and co-author of AutoCAD
    - PGP is available from: http://www.pgpi.org/
    - GnuPG is available from: http://www.gnupg.org/
    - MD5 is available from http://www.fourmilab.ch/md5/ or ftp://coast.cs.purdue.edu/pub/tools/unix/crypto/md5/

- Get and install the latest SSH with server components from http://www.ssh.org
  - You must install SSH on your system. This installation uses the Kerberos V5 header file (/usr/local/include/krb5.h), libraries (libcom_err.a, libkrb5.a, libkrb5crypto.a), and krb5.conf. Some companies may require additional security for kerberos passwords and incorporate a mod_auth_kerb of their own. It will likely need to be copied to the <apache dir>/src/modules directory. Check with your local security folks to be sure.

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

- You may need to rename bstring.h in the /usr/local/include directory to something else to avoid being seen by the Apache configuration script (observed problem with Sun Solaris). The Kerberos distribution provides a copy of bstring.h that incorrectly redefines the memset() prototype. The Apache configuration script looks for header files and assumes every one it finds on the system is correct and attempts to include it in the buff.c code.

- Get the latest Apache from http://www.apache.org/dist/httpd/
  - Untar the Apache source
    - tar xvf apache_x.y.z.tar
- Get the latest OpenSSL from http://www.openssl.org/source/
  - See http://httpd.apache.org/docs-2.0/ssl/ssl_overview.html
  - Build OpenSSL
    - tar xvf openssl-x.y.z.tar
    - # cd openssl-x.y.z
    - # ./config
    - # make
    - # make install
    - # cd ..

- Get the latest mod_ssl from http://www.modssl.org/source/
  - Untar mod_ssl and configure the SSL-aware Apache
    - tar xvf mod_ssl-a.b.c-x.y.z.tar
    - # cd mod_ssl-a.b.c-x.y.z
    - # ./configure \
      --with-apache=../apache_x.y.z \
      --with-ssl=../openssl-x.y.z \
      --prefix=/usr/local/apache
    - # cd ..
  - Modify the SSL-aware Apache configuration
    - # cd apache_x.y.z
    - # cd src
    - # cp Configuration.apaci Configuration
    - Edit the Configuration file for the following:
      # The location of the kerberos library
      EXTRA_LDFLAGS=-L/usr/local/lib
      EXTRA_LIBS=-lcom_err -lkrb5 -lk5crypto
      # The location of the kerberos header files
      EXTRA_INCLUDES=-I/usr/local/include
      AddModule modules/kerb/libkerb.a
    - # cd ../../apache_x.y.z
    - # cd src
    - # Configure
    - # cd ../..
- Make and Install Apache
  - # cd apache_x.y.z
  - # make

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

- Check the httpd binary to see that mod_kerb, mod_ssl, and other appropriate modules are in it by doing an "apache_x.y.z/src/httpd -l". The "-l" is not a negative one. It is phonetically a "dash-el".

  The output should like like this:

  ```
  secureserver_xyz# apache_x.y.z/src/httpd -l
  Compiled-in modules:
    http_core.c
    mod_kerb.c
    mod_env.c
    mod_log_config.c
    mod_mime.c
    mod_negotiation.c
    mod_status.c
    mod_include.c
    mod_autoindex.c
    mod_dir.c
    mod_cgi.c
    mod_asis.c
    mod_imap.c
    mod_actions.c
    mod_userdir.c
    mod_alias.c
    mod_access.c
    mod_auth.c
    mod_setenvif.c
    mod_ssl.c
  ```

- At this point you have successfully made Apache. The next step is to install it. Before you do, if you're going to get a certificate for your server in the future, you may want to:
  - "# make certificate"

    This is not necessary but may be helpful to your installation later when you get your server's certificate. It will make some default entries for SSL certificate usage in your httpd.conf file. It won't disrupt anything and can only help you. It's not that difficult, but the installation of a certificate for your server is beyond the purpose of this paper. The certificate basically means the server is who it claims to be, assuming the issuing authority of the certificate is who it claims to be. When you use a check or credit card to purchase something, most stores require an additional piece of identification. Like your driver's license from your state's motor vehicle department helps tell who you are, a certificate from a credible certifying authority vouches for your computer's identification. The

© SANS Institute 2009, Author retains full rights.

scope of this paper is more about the "driving" and less about the "license".

- Finally, do a:
  - # make install

- Edit the httpd.conf file to be kerberos aware
  - In "Section 1: Global Environment", after the line "ServerType standalone":

```
KrbAuthCache On
KrbLog logs/kerberos_log
```

  - In the "SSL Support" paragraph, modify the listening ports. Port 80 for http and 443 for https are standard numbers for these ports, however you can make them different.

```
##  SSL Support
##
##  When we also provide SSL we have to listen to the
##  standard HTTP port (see above) and to the HTTPS port
##
<IfDefine SSL>
        Listen 80
        Listen 443
</IfDefine>
```

  - You may restrict access to your server to particular IPs or DNS domains. I would recommend you restrict the https access to only those individuals needing access, based on their kerberos login only. This gives the user the flexibility to authenticate from any machine on your network, without preloading Apache's authentication mechanism with users. Apache's authentication mechanism is far inferior to a Kerberos login anyway. Put in the httpd.conf for a directory restriction (example):

```
<Directory />
        Options Indexes FollowSymLinks MultiViews
        #AllowOverride None
        AllowOverride AuthConfig
        # You can substitute your custom short string for "Secure-
Kerberos".
        # It is the text in a dialogue box that prompts the user for their
password
        # and usually defines the Kerberos authentication realm
        AuthName Secure-Kerberos
        AuthType KerberosV5
        require valid-user
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
        Order allow,deny
        Allow from all
    </Directory>
```

- For the sake of simplicity, make ".htaccess" the AccessFileName and restrict it's viewing using the directive:

```
    <Files ~ "^\.ht">
      Order allow,deny
      Deny from all
    </Files>
```

- In httpd.conf, edit the CustomLog directive section to include the line:

```
    LogFormat "%{%D%T}t %h %u \"%r\" %>s %{KerbAuth}n"
    Kerberos
```

- You are strongly encouraged to run the Apache server as a non-root user. The example here shows the account "wwwuser" with it's own group of "www". You could use a restricted shell (http://www.securityfocus.com/tools/590) for the non-root user or use suexec (http://httpd.apache.org/docs-2.0/suexec.html), which allows CGI programs to be run with different user permissions. It is good practice to only put the web account in the web group. However, at a minimum, make an entity account with a locked out password field (use an asterisk in the password field of the /etc/passwd or /etc/shadow) and add the lines to httpd.conf:

```
User wwwuser
Group wwwgroup
```

- Start the server. Use the "startssl" parameter in start script, usually located in /etc/rc2.d/S98apache. For example:
  - /usr/local/apache/bin/apachectl startssl

- Connect to your server from a client browser using the "https" prefix. You should be prompted for your Kerberos password in order to gain access. For example:
  - https://secureserver_xyz/

Section II: Set Up CGI Scripts

Overview

This section's objective is to set up appropriate directories with appropriate access controls to the account running the web server and to the user authenticating via the web server. It uses some interesting scripts which allow the selection of a password, which is stored in encrypted form in a protected

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

directory. The contents of the encrypted file are then read by a privelged account, which modifies the password for the specified user.

Tasks

- Set Up cgi-bin/password CGI scripts
  - You will need to obtain Perl (http://www.perl.org) which is used to obtain information from invalid access to the web password form. Typically and for the purposes of this paper, it is installed in /usr/local/bin.
  - Enable files with extensions of "cgi" to be run from your web server, assuming your ScriptAlias variable in httpd.conf is set to /usr/local/apache/cgi-bin (the Apache default):
    - In the "<IfModule mod_alias.c>" section of your httpd.conf, ensure the directive is present:

```
<Directory "/usr/local/apache/cgi-bin">
      SSLOptions +StdEnvVars
</Directory>

<Directory "/usr/local/apache/cgi-bin">
   AllowOverride AuthConfig
   #AllowOverride None
   Options None
   Order allow,deny
   Allow from all
</Directory>

<Files ~ "\.(cgi|shtml|phtml|php3?)$">
   SSLOptions +StdEnvVars
</Files>
```

  - Make the directory /usr/local/apache/cgi-bin/passwd. Set the permissions on this directory are 0755, owned by root and grouped to bin. The log files kept by these scripts will not contain any password information. Combined with Apache's log files, produce an accurate record of a user Web Password Page access and script actions. The scripts use the "post" method of transferring data to one another. As opposed to the "get" method, the "post" method is a bit more complex to use and allows that data to be separate from the URL. It's used here to prevent users from using URL arguments to compromise data or even see how information is being passed around. The variables are "posted" internal to the process on the Apache server, separate from the memory process on the browser client end. Deposit the following scripts, also with the same permissions in this directory:
    - pwform.cgi:

```
#!/bin/sh
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
##############################################################################
#
#  Program:              /usr/local/apache/cgi-bin/password/pwform.cgi
#  Credit:   In God We Trust
#
#  Purpose:  The Web Password Form:
#                                    1)  Dynamically generate a user page in memory.
#                                    2)  Redirect the browser to load the page in the server's memory.
#
#      OS:  IRIX, IRIX64
#
##############################################################################

#########################################################################
# Variable setup
#########################################################################
UPDATE_NAME="Web Password Form:  Generate user page in memory.  Transfer user to it."
SCRIPT_SHORTNAME="pwform.cgi"
SCRIPT_VERSION=1.0
LOGDATE=`/usr/bin/date +%d%h%y-%T`
# Make the LOGFILE directory mode 500 and owned by the web account
LOGFILE=/usr/local/apache/logs/harvest/form.log
DAYMONTHYEAR=`/usr/bin/date +%d%h%y`

#########################
# Independent Variables #
#########################
AWK=/bin/awk
CAT=/usr/bin/cat
CD=cd
CHGRP=/usr/bin/chgrp
CHGRP=/usr/bin/chgrp
CHMOD=/usr/bin/chmod
CHOWN=/usr/bin/chown
CP=/usr/bin/cp
CUT=/usr/bin/cut
DATE=/usr/bin/date
ECHO=/usr/bin/echo
EGREP=/usr/bin/egrep
EXPR=/sbin/expr
GREP=/usr/bin/grep
MAIL=/usr/sbin/Mail
MKDIR=/usr/bin/mkdir
MV=/usr/bin/mv
PS=/usr/bin/ps
RM=/bin/rm
SED=/usr/bin/sed
SLEEP=/bin/sleep
SORT=/usr/bin/sort
TAR=/usr/bin/tar
TEE=/usr/bin/tee
TOUCH=/usr/bin/touch
TR=/usr/bin/tr
UNIQ=/usr/bin/uniq
WC=/sbin/wc

#######################
# Dependent Variables #
#######################
MINUTES=`$ECHO $LOGDATE | $AWK -F: '{ print $2 }'`
OS=`/bin/uname -s | $CUT -c1-4`
OSR=`/bin/uname -r | $CUT -d. -f1`
OSRR=`/bin/uname -r | $CUT -d. -f2`
SYS_NAME=`/sbin/uname -s | $CUT -c1-4`

#############################
# Script Specific Variables #
#############################
ADMIN_EMAIL="webadmin@servername_xyz.com"
```

© SANS Institute 2009, Author retains full rights.

```
##########################################################################
# Configure mail message (optional)
##########################################################################
UPDATE_DESC="Description:  ${UPDATE_NAME}"
##########################################################################

##########################################
#Subroutine Declarations
##########################################
# lecho - Logged echo to stdout
#     Arg 0 to N = Data to be echoed
lecho()
{
        $ECHO "$REMOTE_USER:      $@" >> $LOGFILE
        $ECHO "$@"
}

echolog()
{
        $ECHO "$REMOTE_USER:      $@" >> $LOGFILE
}

$TOUCH $LOGFILE
echolog "#################### BEGIN Script:  $SCRIPT_SHORTNAME, $LOGDATE"
echolog "$UPDATE_DESC"

ARGS1=`$TR "&+" " "`
ARGS=`$ECHO $ARGS1 | $SED s@%2F@/@g`
if [ -n "$QUERY_STRING" -o -n "$ARGS" -o -z "$REMOTE_USER" ]; then
        $ECHO "<center>"
        lecho "$LOGDATE    Your wrongful attempt to access this page was logged.<br>Appropriate personnel will be notified."
        $ECHO "</center>"
        echolog "QUERY_STRING=$QUERY_STRING"
        echolog "ARGS=$ARGS"
        echolog "Environment follows:"
        /usr/local/bin/perl /usr/local/apache/cgi-bin/password/getenv.pl | $TEE -a $LOGFILE | $MAIL -s "$SCRIPT_SHORTNAME -
Wrongful Attempt to Access Web Password Change Form" $ADMIN_EMAIL
        exit 1
fi

echolog "Accessing password change form."

$CAT << EOF

<head><title>User Page:  Change Your Password</title><meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<center>
<h3>Web Password Change Form for '$REMOTE_USER'</h3>
</center>
<table cellspacing=5>
</table><p>
<blockquote><b>Directions:</b><blockquote>
1)  Choose a new password.<br>
2)  Enter it twice (once in each box).<br>
3)  Check your selections and click on the <i>Submit New Password</i> button at the bottom of this page.
</blockquote></blockquote>
<center>

<form action="https://servername_xyz/cgi-bin/password/pwform.cgi" method="post">
<input type="submit" value="Generate Different Passwords">
</form>

<form action ="https://servername_xyz/cgi-bin/password/pwupdate.cgi" method="POST" name="password">
<table cellspacing=10 border=4>

EOF

COUNT=0
OPTIONS="-c -d -m -C -D -M"
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
for OPTION in ""$OPTIONS""; do
        GETMORE="yes"
        while [ "$GETMORE" = "yes" ]; do
                PASSWORDS=`/usr/local/bin/gen_password $OPTION -P 5`
                $ECHO $PASSWORDS | egrep -s "1|0" > /dev/null 2>&1
                if [ $? -ne 0 ]; then
                        GETMORE="no"
                fi
        done
        $ECHO "<tr>"
        for PASSWORD in ""$PASSWORDS""; do
                COUNT=`$EXPR $COUNT + 1`
                $ECHO "<td><b><tt>$PASSWORD</tt></b></td>"
                $ECHO "<input type="hidden" name="PASSWORD_CHOICE$COUNT" value=$PASSWORD>"
        done
        $ECHO "</tr>"
done

$CAT << EOF
</table>
<table cellspacing=5>
<blockquote>
<blockquote>
<blockquote>
<b>
<TT> <font color="009900"><b>NOTE:</b></font> "l" is a letter and "O" is a letter.<br><b><u>None</u></b> of these choices contain the
digit 1 or the digit 0.</tt></b>
</blockquote>
</blockquote>
</blockquote>

<tr><td>Enter your <b><i>new password</i></b> once: </td><td><input type=password name=NEWPASSWORD1 size=8 maxlength=8
></td></tr>
<tr><td>Enter your <b><i>new password</i></b> a second time: </td><td><input type=password name=NEWPASSWORD2 size=8
maxlength=8 ></td></tr>
<tr><td> </td><td><input type="reset" value="Reset All"></td></tr>
</table>

<blockquote>
<font color=RED><b>WARNING:</b></font>  The password you choose will not be displayed again.<br>
If you don't already have it committed to memory, write it down <b>NOW!</b>

<br><P>
<table cellspacing=0>
</table>
<br><input type=submit name=SUBMIT_BUTTON value="Submit Password">
</form>
</blockquote>
</center>

<div align="right">
<font size="-2">Web Password Page<br>Ver. 1.0, February 2001</font>
</div>
</body></html>

EOF

$TOUCH $LOGFILE

echolog "#################### END Script:  $SCRIPT_SHORTNAME"

exit 0
```

- This file has been re-named to pwform.txt for the purposes of viewing and not executing it.
- It is the first page to load. The URL for clients will be https://secureserver_xyz/cgi-bin/password/pwform.cgi.

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

- Place this file (be sure to dos2unix it in order to remove the hard returns from the ascii code) in the cgi-bin/passwd directory and rename it to "pwform.cgi".
- It uses an IRIX binary to generate passwords. Various options give more different types of passwords for the user to select from. I've chosen to parse through the password specification options "-c -d -m -C -D -M", using each one type for each row (6 rows, 5 columns). Although you can use just about any password generator, gen_password[1] has several options (see also the output example below, "User Password Change Form") including:

  ./gen_password -h

  Usage:
  gen_password [-h | -c | -d | -m | -C | -D | -M] [-P #]
  Generate 8 Character Pseudo-Pronouncible Passwords with
  Characteristics defined by one of the flags.

  Option ==> Meaning:
  h ==> Print this list
  Password Specifications:
  c ==> 1 Uppercase, 7 lowercase, no digits
  d ==> 1 Uppercase, 1-7 lowercase, 0-6 digits
  m ==> 1 Uppercase, 1-6 lowercase, 1-6 digits
  C ==> 1-7 Uppercase, 1-7 lowercase, no digits
  D ==> 1-7 Uppercase, 1-7 lowercase, 0-6 digits
  M ==> 1-6 Uppercase, 1-6 lowercase, 1-6 digits
  A ==> 0-8 Uppercase, 0-8 lowercase, 0-8 digits
  P ==> Password Count / Loop (-L #)
  1-1000 (Default = 1)

  [1] Rahe, Bill, and others. "gen_password" Jan. 1998. URL: http://www.lanl.gov/asci/DCE/Presentations/Workshop-Jan98/dce_accounts.ppt

- pwupdate.cgi:

```
#!/bin/sh
##############################################################################
#
#  Program:            /usr/local/apache/cgi-bin/password/pwupdate.cgi
#  Credit:   In God We Trust
#
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
#  Purpose:  The Web Password Form:  Update data directory with password for user.
#
#       OS:  IRIX, IRIX64
#
################################################################################

##########################################################################
# Variable setup
##########################################################################
UPDATE_NAME="Web Password Form:  Update data directory with password for user."
SCRIPT_SHORTNAME="pwupdate.cgi"
SCRIPT_VERSION=1.0
LOGDATE=`/usr/bin/date +%d%h%y-%T`
# Make the LOGFILE directory mode 500 and owned by the web account
LOGFILE=/usr/local/apache/logs/harvest/form.log
DAYMONTHYEAR=`/usr/bin/date +%d%h%y`

#########################
# Independent Variables #
#########################
AWK=/bin/awk
CAT=/usr/bin/cat
CD=cd
CHGRP=/usr/bin/chgrp
CHGRP=/usr/bin/chgrp
CHMOD=/usr/bin/chmod
CHOWN=/usr/bin/chown
CP=/usr/bin/cp
CRYPT=/usr/bin/crypt
CUT=/usr/bin/cut
DATE=/usr/bin/date
ECHO=/usr/bin/echo
EGREP=/usr/bin/egrep
EXPR=/sbin/expr
GREP=/usr/bin/grep
MAIL=/usr/sbin/Mail
MKDIR=/usr/bin/mkdir
MV=/usr/bin/mv
PS=/usr/bin/ps
RM=/bin/rm
SED=/usr/bin/sed
SLEEP=/bin/sleep
SORT=/usr/bin/sort
TAR=/usr/bin/tar
TEE=/usr/bin/tee
TOUCH=/usr/bin/touch
TR=/usr/bin/tr
UNIQ=/usr/bin/uniq
WC=/sbin/wc

#######################
# Dependent Variables #
#######################
MINUTES=`$ECHO $LOGDATE | $AWK -F: '{ print $2 }'`
OS=`/bin/uname -s | $CUT -c1-4`
OSR=`/bin/uname -r | $CUT -d. -f1`
OSRR=`/bin/uname -r | $CUT -d. -f2`
SYS_NAME=`/sbin/uname -s | $CUT -c1-4`

#############################
# Script Specific Variables #
#############################
ADMIN_EMAIL="webadmin@servername_xyz.com"
IDEA=/bin/idea
DATADIR=/opt/securewebdata/passwords
PASSWORD_LENGTH_MAX=8
PUBLICKEY=`$CAT /opt/securewebdata/keys/public.unix`
PRIVATEKEYFILE=/opt/securewebdata/keys/private.unix
PRIVATEKEY=`$IDEA -d -k $PUBLICKEY $PRIVATEKEYFILE`
# WWWUSER and WWWGROUP are the account and group that runs the https server.
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
WWWUSER="wwwuser"
WWWGROUP="wwwgroup"

##########################################################################
# Configure mail message (optional)
##########################################################################
UPDATE_DESC="Description:  ${UPDATE_NAME}"
##########################################################################

##########################################
#Subroutine Declarations
##########################################
# lecho - Logged echo to stdout
#     Arg 0 to N = Data to be echoed
lecho()
{
        $ECHO "$REMOTE_USER:      $@" >> $LOGFILE
        $ECHO "$@"
}

echolog()
{
        $ECHO "$REMOTE_USER:      $@" >> $LOGFILE
}

##########################################
# nuke - a kinder, gentler obliterator
nuke()
{       if [ -n "$1" -a "$1" != "/" ]; then
                $RM -fr "$1"
        fi
}

##########################################
# own_grp_mod -       Changes user and group ownership and modifies permissions
own_grp_mod()
{
if [ -n $1 -a -n $2 -a -n $3 -a -n $4 ];then
        if [ -f "$4" -o -d "$4" ];then
                OLD_OBJECTDATA=`ls -ld $4`
                echolog "OLD:  $OLD_OBJECTDATA"
                $CHOWN $1 $4
                $CHGRP $2 $4
                $CHMOD $3 $4
                NEW_OBJECTDATA=`ls -ld $4`
                echolog "NEW:  $NEW_OBJECTDATA"
        else
                echolog "File or directory \"${4}\" does not exist."
        fi
else
        echolog "Parameter missing, did not change permissions for \"${4}\"."
fi
}

##########################################
# makedirectory
makedirectory()
{       if [ ! -d "$1" ];then
        $MKDIR -p "$1"
        fi
}

##########################################
# invalid_access
log_invalid_access()
{       lecho "$LOGDATE    Your wrongful attempt to access this page was logged.<br>Appropriate personnel will be notified."
        echolog "QUERY_STRING=$QUERY_STRING"
        echolog "ARGS=$ARGS"
        echolog "Environment follows:"
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
	/usr/local/bin/perl /usr/local/apache/cgi-bin/password/getenv.pl | $TEE -a $LOGFILE | $MAIL -s "$SCRIPT_SHORTNAME -
Wrongful Attempt to Access Web Password Change Form" $ADMIN_EMAIL
	exit 1
}

# disable filename globbing
set -f

$ECHO "Content-type: text/html"
$ECHO ""

$TOUCH $LOGFILE
echolog "#################### BEGIN Script:  $SCRIPT_SHORTNAME, $LOGDATE"
echolog "$UPDATE_DESC"

NUMBER=0
echolog "Result of User Web Password Change Form"
$ECHO "<head><title>Result of User Web Password Change Form</title><meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1">
</head>"
#read buffer
ARGS1=`$TR "&+" " "`
ARGS=`$ECHO $ARGS1 | $SED s@%2F@/@g`
IFS=" "

$CAT << EOF
<meta name="Description" content="">
<meta name="Keywords" content="">
<meta name="Author" content="Mark Holbrook">
<meta name="Generator" content="Web Password Generator">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<center>
<h2>Results of User Web Password Change Form</h2>
<p>
<blockquote>
<blockquote>

EOF

USERNAME="$REMOTE_USER"
echolog "Attempting password change."
if [ -z "$QUERY_STRING" -a -n "$ARGS" ]; then
	for ITEM in ""$ARGS""; do
		NUMBER=`$EXPR $NUMBER + 1`
		VARNAME=`$ECHO $ITEM | $AWK -F"=" '{ print $1 }'`
		VARVALUE=`$ECHO $ITEM | $AWK -F"=" '{ print $2 }'`
		if [ "$VARNAME" = "NEWPASSWORD1" ]; then
			PASSWORD1="$VARVALUE"
			#$ECHO "ITEM number $NUMBER = $ITEM<br>"
			#$ECHO "VARVALUE = $VARVALUE<br>"
		fi
		if [ "$VARNAME" = "NEWPASSWORD2" ]; then
			PASSWORD2="$VARVALUE"
		fi
	done
	makedirectory $DATADIR/$USERNAME
	own_grp_mod $WWWUSER $WWWGROUP 700 $DATADIR/$USERNAME

	PASSWORD_IN_CHOICES="no"
	if [ "$PASSWORD1" = "$PASSWORD2" ]; then
		PASSWORD="$PASSWORD1"
		PASSWORD_LENGTH=`$ECHO $PASSWORD | $WC -m`

		# Check out password for spaces - possible tampering with input
		COUNT=1
		while [ $COUNT -le $PASSWORD_LENGTH ]; do
			CHAR=`$ECHO "$PASSWORD" | $CUT -c${COUNT},${COUNT}`
			$ECHO "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890" | $GREP -s
"$CHAR" > /dev/null
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
                        if [ $? -ne 0 ]; then
                          log_invalid_access
                        fi
                        COUNT=`expr $COUNT + 1`
                done

                PASSWORD_LENGTH=`$EXPR $PASSWORD_LENGTH - 1`
                if [ -n "$PASSWORD" ]; then
                        if [ "$PASSWORD_IN_CHOICES" = "yes" ]; then
                                if [ $PASSWORD_LENGTH -eq $PASSWORD_LENGTH_MAX ]; then
                                        lecho "Your password has been entered successfully."
                                        $ECHO "<br>"
                                        $ECHO $PASSWORD | $IDEA -e -k $PRIVATEKEY >
"$DATADIR/$USERNAME/${USERNAME}.unix"
                                        $ECHO "[data]" > $DATADIR/$USERNAME/${USERNAME}.txt
                                        $ECHO "NewPassword=$PASSWORD" >>
$DATADIR/$USERNAME/${USERNAME}.txt
                                        $IDEA -e -k $PRIVATEKEY $DATADIR/$USERNAME/${USERNAME}.txt
$DATADIR/$USERNAME/${USERNAME}.nt
                                        nuke $DATADIR/$USERNAME/${USERNAME}.txt
                                        own_grp_mod $WWWUSER $WWWGROUP 750
$DATADIR/$USERNAME/${USERNAME}.unix
                                        own_grp_mod $WWWUSER $WWWGROUP 750
$DATADIR/$USERNAME/${USERNAME}.nt
                                else
                                        lecho "Your password must be $PASSWORD_LENGTH_MAX characters long."
                                        $ECHO "<br>"
                                        lecho "Your password was NOT updated."
                                        $ECHO "<br>"
                                fi
                        else
                                lecho "You must choose a password within the given choices."
                                $ECHO "<br>"
                                lecho "Your choice did not match any of the choices provided."
                                $ECHO "<br>"
                                lecho "Your password was NOT updated."
                                $ECHO "<br>"
                        fi
                else
                        lecho "You did not enter a password."
                        $ECHO "<br>"
                        lecho "Your password was NOT updated."
                        $ECHO "<br>"
                fi
        else
                lecho "Your password entries did not match."
                $ECHO "<br>"
                lecho "Your password was NOT updated."
                $ECHO "<br>"
        fi

$CAT << EOF
</blockquote>
</blockquote>
<p>
<center>
<script language="JavaScript">
<!-- Begin
var name = navigator.appName
url=(document.referrer)
document.write('<A HREF="' + url + '">Go Back</A>');
// End -->
</script>
</center>
<p>
<div align="right">
<font size="-2">Web Password Page<br>Ver. 1.0, February 2001</font>
</div>

EOF
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
else
        log_invalid_access
fi

$TOUCH $LOGFILE

echolog "#################### END Script:  $SCRIPT_SHORTNAME"

exit 0
```

- This file has been re-named to pwform.txt for the purposes of viewing and not executing it.
- Place this file (be sure to dos2unix it in order to remove the hard returns from the ascii code) in the cgi-bin/passwd directory and rename it to "pwupdate.cgi".
- This script security takes data entered via the web form and stores it in a protected file on the server.
- You will need to review the variables used in the script to customize it for your use (example output, "Results of User Password Change Form" pictured below).

- getenv.pl
  - This is a perl script which gathers information about a client in order to log invalid accesses to the password form.
  - Perl seems to do a very nice job at detecting and collecting the browser data.
  - Contents of the getenv.pl file (this script came packaged with my Apache source):

```
#!/usr/local/bin/perl
##
##  printenv -- demo CGI program which just prints its
environment
##
print "Content-type: text/plain\n\n";
foreach $var (sort(keys(%ENV))) {
   $val = $ENV{$var};
   $val =~ s|\n|\\n|g;
   $val =~ s|"|\\"|g;
   print "${var}=\"${val}\"\n";
}
```

- Set Up the Password Data Harvest Mechanism
  - You will also need to get IDEA, (International Data Encryption Algorithm, ftp://ftp.cert.dfn.de/pub/tools/crypt/idea), or another strong symmetric key cipher.
  - You will also need to install Expect. Expect is a wrapper for just about any command needing stdin. Expect (http://expect.nist.gov) for IRIX can be found at http://freeware.sgi.com.

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

- Be sure to set the securewebdata (see the pwupdate.cgi script) to permissions: root owned, sys grouped, chmod 755.
- Be sure to set the securewebdata/passwords subdirectory permissions: web account owned, webgroup grouped, chmod 500.
- Make the directory securewebdata/keys, and set the permissions to: root owned, webgroup grouped, chmod 550. This is the directory where the "private" and "public" Windows and UNIX encrypted and unencrypted files will reside (root owned, webgroup grouped and chmod 440). Of course these "public" and "private" keys are strictly by PKI (see http://www.pki-page.org) definition, but the concept does loosely apply. The "public" key is not encrypted, while the "private" one is and is more closely guarded.
- Make the directory securewebdata/bin, and set the permissions to: root owned, bin grouped, chmod 500. This is where the scripts run by root will gather the data deposited by the web account and modify the user passwords.
- Make the expect script expectpass.sh, and set permissions to root owned, bin grouped, chmod 500:

```
#!/usr/bin/expect
# this script called with username and password args, respectively
set password [lindex $argv 1]
spawn /bin/passwd [lindex $argv 0]
expect "assword*:"
send "$password\r"
expect "assword*:"
send "$password\r"
expect eof
exit 0
```

- The Cipher Key Files
  - The key files are just another way to implement an added layer of security to the password harvest data mechanism. Simply put, a "public" file contains an unencrypted password in plain text that is used by the IDEA cipher to decrypt the contents of the "private" file (the contents of these "public and "private" key files must be set manually before-hand). This content is the private password used to decrypt the password data files deposited by the web password forms. The password is fed to the expect script, thereby changing the user's password. During all of this no temp files are used. No user's password resides on the disk unencrypted. No sensitive password information is written to disk, uncrypted or encrypted, except for the expect script using the /bin/passwd binary to write the encrypted password to the /etc/shadow file. It is all done in protected memory, running as root. This minimizes the necessity of viewing the password while manually running the /bin/passwd command, thereby exposing it to the administrator. Granted the

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

administrator has access to the data, but is not required to view the actual password, minimizing password exposure.

- Examples
  - #cat securewebdata/keys/public.unix
    passwordinplaintext
  - #cat securewebdata/keys/private.unix
    Tf ╒q.↨╧tN⌂⌂A▒k│q
  - #cat securewebdata/keys/public.nt
    [data]
    password="passwordinplaintext"
  - #cat securewebdata/keys/private.nt
    ²1C╥í$│1ö--ì_ì╡êal3W{}G|+mù≥!!V`ç½╕Öxg/-

- Use a Script to Automatically Update the Passwords
  - UNIX
    Overview: Use "pwharvest.sh" to gather the data from the private files and populate the local authentication mechanism (/etc/shadow).
    - pwharvest.sh:

```
#!/bin/sh
##############################################################################
#
#  Program:              /usr/local/apache/cgi-bin/password/pwupdate.cgi
#  Credit:   In God We Trust
#
#  Purpose:  The Web Password Form:  Update data directory with password for user.
#
#      OS:  IRIX, IRIX64
#
##############################################################################

#########################################################################
# Variable setup
#########################################################################
UPDATE_NAME="Web Password Form:  Update data directory with password for user."
SCRIPT_SHORTNAME="pwupdate.cgi"
SCRIPT_VERSION=1.0
LOGDATE=`/usr/bin/date +%d%h%y-%T`
# Make the LOGFILE directory mode 500 and owned by the web account
LOGFILE=/usr/local/apache/logs/harvest/form.log
DAYMONTHYEAR=`/usr/bin/date +%d%h%y`

#########################
# Independent Variables #
#########################
AWK=/bin/awk
CAT=/usr/bin/cat
CD=cd
CHGRP=/usr/bin/chgrp
CHGRP=/usr/bin/chgrp
CHMOD=/usr/bin/chmod
CHOWN=/usr/bin/chown
CP=/usr/bin/cp
CRYPT=/usr/bin/crypt
CUT=/usr/bin/cut
DATE=/usr/bin/date
ECHO=/usr/bin/echo
EGREP=/usr/bin/egrep
EXPR=/sbin/expr
GREP=/usr/bin/grep
MAIL=/usr/sbin/Mail
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
MKDIR=/usr/bin/mkdir
MV=/usr/bin/mv
PS=/usr/bin/ps
RM=/bin/rm
SED=/usr/bin/sed
SLEEP=/bin/sleep
SORT=/usr/bin/sort
TAR=/usr/bin/tar
TEE=/usr/bin/tee
TOUCH=/usr/bin/touch
TR=/usr/bin/tr
UNIQ=/usr/bin/uniq
WC=/sbin/wc

######################
# Dependent Variables #
######################
MINUTES=`$ECHO $LOGDATE | $AWK -F: '{ print $2 }'`
OS=`/bin/uname -s | $CUT -c1-4`
OSR=`/bin/uname -r | $CUT -d. -f1`
OSRR=`/bin/uname -r | $CUT -d. -f2`
SYS_NAME=`/sbin/uname -s | $CUT -c1-4`

############################
# Script Specific Variables #
############################
ADMIN_EMAIL="webadmin@servername_xyz.com"
IDEA=/bin/idea
DATADIR=/opt/securewebdata/passwords
PASSWORD_LENGTH_MAX=8
PUBLICKEY=`$CAT /opt/securewebdata/keys/public.unix`
PRIVATEKEYFILE=/opt/securewebdata/keys/private.unix
PRIVATEKEY=`$IDEA -d -k $PUBLICKEY $PRIVATEKEYFILE`
# WWWUSER and WWWGROUP are the account and group that runs the https server.
WWWUSER="wwwuser"
WWWGROUP="wwwgroup"

##########################################################################
# Configure mail message (optional)
##########################################################################
UPDATE_DESC="Description:  ${UPDATE_NAME}"
##########################################################################

##########################################
#Subroutine Declarations
##########################################
# lecho - Logged echo to stdout
#     Arg 0 to N = Data to be echoed
lecho()
{
        $ECHO "$REMOTE_USER:      $@" >> $LOGFILE
        $ECHO "$@"
}

echolog()
{
        $ECHO "$REMOTE_USER:      $@" >> $LOGFILE
}

##########################################
# nuke - a kinder, gentler obliterator
nuke()
{       if [ -n "$1" -a "$1" != "/" ]; then
                $RM -fr "$1"
        fi
}

##########################################
# own_grp_mod -       Changes user and group ownership and modifies permissions
own_grp_mod()
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
{
if [ -n $1 -a -n $2 -a -n $3 -a -n $4 ];then
        if [ -f "$4" -o -d "$4" ];then
                OLD_OBJECTDATA=`ls -ld $4`
                echolog "OLD:  $OLD_OBJECTDATA"
                $CHOWN $1 $4
                $CHGRP $2 $4
                $CHMOD $3 $4
                NEW_OBJECTDATA=`ls -ld $4`
                echolog "NEW:  $NEW_OBJECTDATA"
        else
                echolog "File or directory \"${4}\" does not exist."
        fi
else
        echolog "Parameter missing, did not change permissions for \"${4}\"."
fi
}

##########################################
# makedirectory
makedirectory()
{       if [ ! -d "$1" ];then
        $MKDIR -p "$1"
        fi
}

##########################################
# invalid_access
log_invalid_access()
{       lecho "$LOGDATE    Your wrongful attempt to access this page was logged.<br>Appropriate personnel will be notified."
        echolog "QUERY_STRING=$QUERY_STRING"
        echolog "ARGS=$ARGS"
        echolog "Environment follows:"
        /usr/local/bin/perl /usr/local/apache/cgi-bin/password/getenv.pl | $TEE -a $LOGFILE | $MAIL -s "$SCRIPT_SHORTNAME -
Wrongful Attempt to Access Web Password Change Form" $ADMIN_EMAIL
        exit 1
}

# disable filename globbing
set -f

$ECHO "Content-type: text/html"
$ECHO ""

$TOUCH $LOGFILE
echolog "#################### BEGIN Script:  $SCRIPT_SHORTNAME, $LOGDATE"
echolog "$UPDATE_DESC"

NUMBER=0
echolog "Result of User Web Password Change Form"
$ECHO "<head><title>Result of User Web Password Change Form</title><meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1">
</head>"
#read buffer
ARGS1=`$TR "&+" " "`
ARGS=`$ECHO $ARGS1 | $SED s@%2F@/@g`
IFS=" "

$CAT << EOF
<meta name="Description" content="">
<meta name="Keywords" content="">
<meta name="Author" content="Mark Holbrook">
<meta name="Generator" content="Web Password Generator">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<center>
<h2>Results of User Web Password Change Form</h2>
<p>
<blockquote>
<blockquote>
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
EOF

USERNAME="$REMOTE_USER"
echolog "Attempting password change."
if [ -z "$QUERY_STRING" -a -n "$ARGS" ]; then
	for ITEM in ""$ARGS""; do
		NUMBER=`$EXPR $NUMBER + 1`
		VARNAME=`$ECHO $ITEM | $AWK -F"=" '{ print $1 }'`
		VARVALUE=`$ECHO $ITEM | $AWK -F"=" '{ print $2 }'`
		if [ "$VARNAME" = "NEWPASSWORD1" ]; then
			PASSWORD1="$VARVALUE"
			#$ECHO "ITEM number $NUMBER = $ITEM<br>"
			#$ECHO "VARVALUE = $VARVALUE<br>"
		fi
		if [ "$VARNAME" = "NEWPASSWORD2" ]; then
			PASSWORD2="$VARVALUE"
		fi
	done
	makedirectory $DATADIR/$USERNAME
	own_grp_mod $WWWUSER $WWWGROUP 700 $DATADIR/$USERNAME

	PASSWORD_IN_CHOICES="no"
	if [ "$PASSWORD1" = "$PASSWORD2" ]; then
		PASSWORD="$PASSWORD1"
		PASSWORD_LENGTH=`$ECHO $PASSWORD | $WC -m`

		# Check out password for spaces - possible tampering with input
		COUNT=1
		while [ $COUNT -le $PASSWORD_LENGTH ]; do
			CHAR=`$ECHO "$PASSWORD" | $CUT -c${COUNT},${COUNT}`
			$ECHO "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890" | $GREP -s
"$CHAR" > /dev/null
			if [ $? -ne 0 ]; then
			  log_invalid_access
			fi
			COUNT=`expr $COUNT + 1`
		done

		PASSWORD_LENGTH=`$EXPR $PASSWORD_LENGTH - 1`
		if [ -n "$PASSWORD" ]; then
			if [ "$PASSWORD_IN_CHOICES" = "yes" ]; then
				if [ $PASSWORD_LENGTH -eq $PASSWORD_LENGTH_MAX ]; then
					lecho "Your password has been entered successfully."
					$ECHO "<br>"
					$ECHO $PASSWORD | $IDEA -e -k $PRIVATEKEY >
"$DATADIR/$USERNAME/${USERNAME}.unix"
					$ECHO "[data]" > $DATADIR/$USERNAME/${USERNAME}.txt
					$ECHO "NewPassword=$PASSWORD" >>
$DATADIR/$USERNAME/${USERNAME}.txt
					$IDEA -e -k $PRIVATEKEY $DATADIR/$USERNAME/${USERNAME}.txt
$DATADIR/$USERNAME/${USERNAME}.nt
					nuke $DATADIR/$USERNAME/${USERNAME}.txt
					own_grp_mod $WWWUSER $WWWGROUP 750
$DATADIR/$USERNAME/${USERNAME}.unix
					own_grp_mod $WWWUSER $WWWGROUP 750
$DATADIR/$USERNAME/${USERNAME}.nt
				else
					lecho "Your password must be $PASSWORD_LENGTH_MAX characters long."
					$ECHO "<br>"
					lecho "Your password was NOT updated."
					$ECHO "<br>"
				fi
			else
				lecho "You must choose a password within the given choices."
				$ECHO "<br>"
				lecho "Your choice did not match any of the choices provided."
				$ECHO "<br>"
				lecho "Your password was NOT updated."
				$ECHO "<br>"
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
                        fi
                else
                        lecho "You did not enter a password."
                        $ECHO "<br>"
                        lecho "Your password was NOT updated."
                        $ECHO "<br>"
                fi
        else
                lecho "Your password entries did not match."
                $ECHO "<br>"
                lecho "Your password was NOT updated."
                $ECHO "<br>"
        fi

$CAT << EOF
</blockquote>
</blockquote>
<p>
<center>
<script language="JavaScript">
<!-- Begin
var name = navigator.appName
url=(document.referrer)
document.write('<A HREF="' + url + '">Go Back</A>');
// End -->
</script>
</center>
<p>
<div align="right">
<font size="-2">Web Password Page<br>Ver. 1.0, February 2001</font>
</div>

EOF

else
        log_invalid_access
fi

$TOUCH $LOGFILE

echolog "#################### END Script:  $SCRIPT_SHORTNAME"

exit 0
```

- This file has been re-named to pwharvest.txt for the purposes of viewing and not executing it.
- Place this file (be sure to dos2unix it in order to remove the hard returns from the ascii code) in the securewebdata/bin directory, rename it to "pwharvest.sh", and set the permissions to root owned, bin grouped, chmod 500.

- Set Root's Crontab to include (as an extra precaution, pipe the stdout and stderr to the secure bit bucket, /dev/null):

  # Update Passwords
  #
  0 18 * * * /opt/securewebdata/bin/pwharvest.sh > /dev/null 2>&1

- Windows
  Overview: Use "pwharvest_winbatch.wbt" to gather the data from

© SANS Institute 2009, Author retains full rights.

the private files and populate the local authentication mechanism. The method of transferring the private files from the UNIX server to the Windows server is not covered in this paper.

- pwharvest_winbatch.wbt:

```
;
==========================================================================================
;
==========================================================================================
; Program Name:  pwharvest_winbatch.wbt
; ..........................................................................................
; Script Version Number                    Date                    Modification Description          Modified by
; ..........................................................................................
; Purpose:  Extract password data from a plain text file and insert into local accounts
;
==========================================================================================
;
==========================================================================================

; Total DLLs needed:
;           wbdbu32i.dll - bundled with .exe when "large exe" is chosen for compiling
;           wwwnt32i.dll
AddExtender("WWWNT32I.DLL")
AddExtender("wbdbu32i.dll")

;=====================================================================================
;=====================================================================================
;=====================================================================================
; Initialize Variables

ProgramName="pwharvest_winbatch"
ProgramVersion="1.1"
ProgramComments="Matt 9.37-38"

ErrorMode(@CANCEL)
PWD=DirGet( )
SystemRoot=Environment("SystemRoot") ; e.g. C:\WINNT
SystemDrive=Environment("SystemDrive") ; e.g. C:
TimeNow=TimeYmdHms( )
NormalDateTime=TimeDate( )
TimeNowModified=Strreplace (TimeNow,":","-")
LogFileDirectory="%SystemDrive%\web_password_harvest\%ProgramName%\log\"
LogFileName="LogFile_%TimeNowModified%.txt"
LogFile="%LogFileDirectory%%LogFileName%"
LogRegKey="SYSTEM\%ProgramName%"
INIdir="%PWD%ini"
TEMP=Environment("TEMP")
USERNAME=wntGetUser(@DEFAULT)
NetBIOS_ComputerName=RegQueryValue(@REGMACHINE,
"SYSTEM\CurrentControlSet\Control\ComputerName\ActiveComputerName[ComputerName]")
; DateStamp=Timedate( )
; Day of week, mdy, time am/pm
PublicKeyFile=".\keys\public.nt"
PrivateKeyFile=".\keys\private.nt"
PrivateKeyINI=".\keys\private.ini"

;=====================================================================================
;=====================================================================================
;=====================================================================================
; Verify Prerequisites

; Make log file directories
IF !DirExist("%LogFileDirectory%") THEN DirMake("%LogFileDirectory%")

; Record runtime in the registry
RegSetEx(@RegMachine, "%LogRegKey%[Program Name]", "%ProgramName%", ":", "1")
RegSetEx(@RegMachine, "%LogRegKey%[Program Comments]", "%ProgramComments%", ":", "1")
RegSetEx(@RegMachine, "%LogRegKey%[Program Version]", "%ProgramVersion%", ":", "1")
```

© SANS Institute 2009, Author retains full rights.

```
RegSetEx(@RegMachine, "%LogRegKey%[Last run on yyyy-mm-dd-hh-mm-ss]", "%TimeNowModified%", ":", "1")
RegSetEx(@RegMachine, "%LogRegKey%[Run by UserName]", "%USERNAME%", ":", "1")
RegSetEx(@RegMachine, "%LogRegKey%[Log File Name]", "%LogFileDirectory%%LogFileName%", ":", "1")

; Record program version and date in logfile
IniWritePvt("Initialization Data", "%ProgramName% version", "%ProgramVersion%", "%LogFile%")
IniWritePvt("Initialization Data", "current yyyy-mm-dd-hh-mm-ss", "%TimeNowModified%", "%LogFile%")

;==========================================================================================
; Verify OS Type is Windows NT 4.0 or Windows 2000
WinType=WinMetrics(-4)                  ;What type of windows are you running??
WinVersionMajor=WinVersion(1)           ;What is the major build type (Windows 2000 will be "5")
; Check for Administrators group inclusion of account running this script.
IF (%WinType%%WinVersionMajor% == "44") || (%WinType%%WinVersionMajor% == "45")
                ; Windows NT 4.0 or 2000 is running
        ELSE
                Message("ERROR", "This OS is NOT Windows NT 4.0.  Program exiting.")
                GOTO END
ENDIF

;==========================================================================================
; Verify user running script has Administrator priveleges
Result=wntMemberGet("","Administrators","%USERNAME%",@LOCALGROUP)
IF Result == @FALSE
        Message("ERROR","%USERNAME% is not in the Local Administrators Group. Program exiting.")
        GOTO END
ENDIF

; Record username running program
IniWritePvt("Initialization Data", "Username running program with Administrator priveleges", USERNAME,
"%LogFileDirectory%%LogFileName%")

;==========================================================================================
;==========================================================================================
;==========================================================================================

PublicKey=IniReadPvt( "data", "password", "Not Found", PublicKeyFile)
RunShell("cmd.exe", "/c .\idea.exe -d -k %PublicKey% %PrivateKeyFile% %PrivateKeyINI%", ".", @HIDDEN, @WAIT)
PrivateKey=IniReadPvt( "data", "password", "Not Found", PrivateKeyINI)
FileDelete( PrivateKeyINI )

Files=FileItemize("*.nt")
FilesTotal=ItemCount(Files, @TAB)
FOR FileNumber=1 TO FilesTotal
        File=ItemExtract(FileNumber, Files, @TAB)
        UserName=StrSub(File,1,strlen(File)-3)
        FullName=FileFullName(File)
        GoSub ProcessFile
NEXT

GOTO SkipProcessFile
:ProcessFile
DataFile=".\%UserName%.ini"
RunShell("cmd.exe", "/c .\idea.exe -d -k %PrivateKey% %File% %DataFile%", ".", @HIDDEN, @WAIT)
SectionName="Data"
DataNumber=0
NewPassword="something"
NewPassword=IniReadPvt( "Data", "NewPassword", "Not Found", DataFile)
LogText="Error encountered while changing password for account '%name%'"
IF "%NewPassword%" != "Not Found"
        ;NOTE:  Use *UNKNOWN* when you want overwrite existing password, without needing to know the current password
        Result=wntChgPswd("", "%UserName%", "*UNKNOWN*", "%NewPassword%")
        ;message("changing password","new password for '%UserName%' is '%NewPassword%'")
        IF Result
                ;Message("%ProgramName%","Successfully changed password for account '%name%'")
                LogText="Password changed for account '%UserName%'"
        ELSE
                Message("ERROR:  Password Change for '%name%'","Error encountered while changing password for account
'%name%'")
        ENDIF
```

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

```
ELSE
        Message("ERROR:  Password Change for '%name%'","Error encountered while changing password for account '%name%'")
ENDIF
IniWritePvt("Change Password Results", UserName, LogText, "%LogFileDirectory%%LogFileName%")
FileDelete( DataFile )

RETURN

:SkipProcessFile

;==========================================================================================
;==========================================================================================
;==========================================================================================

:END

TimeNowModified=Strreplace (TimeNow,":","-")
IniWritePvt("Change Password Results", "%ProgramName% completed mm-dd-yy-hh-mm", "%TimeNowModified%",
"%LogFileDirectory%%LogFileName%")
; To force the ini updates to disk, add an additional IniWritePvt statement as follows:
IniWritePvt("","","","%LogFileDirectory%%LogFileName%")

Message("%ProgramName% version %ProgramVersion% completed","%@CRLF%The log file is located at
%LogFileDirectory%%LogFileName%.")
```

- This is just one example of how to extract the data from the private file and update a user's password with the unencrypted contents.
- You can get Winbatch from www.winbatch.com.

Section III: Example Screen Shots: What does it look like?

Overview

This section provides examples of what the Web Password Page looks like.

Examples

- The Initial Login Screen - authenticates user. Note that kerberos is used here. The text in this box is configurable via your Apache httpd.conf file.

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

- User selects a password (pwform.cgi html stdout).

User Password Change Form for 'kelli'

Directions:
1) Choose a new password.
2) Enter it twice (once in each box).
3) Check your selections and click on the *Submit New Password* button at the bottom of this page.

Fiuvyiwz	Tyuqgorf	oiftonHu	ziaCmaeh	moegDabg
Veisriwz	obdaEnty	kaorqutT	jvAebwaq	iqryqLia
75Axkaaq	4evXyujc	2obLoebt	ep32faR7	86ipBuey
oiWcuvdo	vaeHkEft	esjOIXHo	ASQOGpae	WiOkRUjR
AU9b4U3U	wGAOwvOk	BOoQWASG	vpiAjzYp	LHIOFwiH
2ubviydb	82eQYAJb	PoQeNOX9	48aPbEyP	3oMCEEkg

NOTE: "l" is a letter and "O" is a letter.
None of these choices contain the digit 1 or the digit 0.

Enter your *new password* once:

Enter your *new password* a second time:

WARNING: The password you choose will not be displayed again.
If you don't already have it committed to memory, write it down now.

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

Web Password Page
Ver. 1.0, February 2001

- Successful submission of password (pwupdate.cgi html stdout)

Results of User Password Change Form

Your password has been entered successfully.

Go Back

Web Password Page
Ver. 1.0, February 2001

Section IV: Summary

The Web Password Page allows the user to change their password local to a UNIX or Windows server, authenticating through the user's Kerberos domain login password (assuming there is one). You could also apply the core concepts of ths paper to update a Kerberos server, Windows Domain Controller, UNIX NIS or NIS+, or other such server's authentication database. Your job as a system administrator is primarily to make the computer do the work. It is the tool and the product of your efforts. You can do it with a little know-how, a couple of popular high-energy software tools, and little "elbow grease". The Web Password Page is a pre-emptive strike approach to attacking the administrative overhead of managing user passwords. It's engine is built on the idea of automating the mundane. Isn't this one of the computer's primary tasks anyway? As the industrial age alleviated mankind's manual labor tasks, so the information age frees up mankind's mental capacities for more rewarding pursuits.

Section V: Bibliography

Citations/References/Resources

- Coar, Ken A. L. Apache Server for Dummies. IDG Books Worldwide, Inc., 1998. 104-105, 207.
- December, John and Ginsburg, Mark. HTML 3.2 and CGI Professional Reference Edition UNLEASHED (26 Jan 2002). Sams.net Publishing, 1996. 335-358.
- Arthur, Lowell Jay and Burns, Ted. UNIX Shell Programming, 4th Edition. Wiley Computer Publishing, John Wiley and Sons, Inc., 1997. 289-295.

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

- Engeschall, Ralf S. "Module mod_ssl." Apache HTTP Server Version 2.0. URL: http://httpd.apache.org/docs-2.0/mod/mod_ssl.html (26 Jan 2002)
- Engeschall, Ralf S. "SSL Module Architecture Overview." mod_ssl 2.8, User Manual, The Apache Interface to OpenSSL, @1998-2001. URL: http://httpd.apache.org/docs-2.0/ssl/ssl_overview.html (26 Jan 2002)
- "Dynamic Content with CGI." Apache HTTP Server Version 2.0. URL: http://httpd.apache.org/docs-2.0/howto/cgi.html (26 Jan 2002)
- "Security Tips for Server Configuration." Apache HTTP Server Version 2.0. URL: http://httpd.apache.org/docs-2.0/misc/security_tips.html (26 Jan 2002)
- "Apache Directives." Apache HTTP Server Version 2.0. URL: http://httpd.apache.org/docs-2.0/mod/directives.html (26 Jan 2002)
- Coar, Ken. "Using .htaccess Files with Apache." Apache HTTP Server Version 2.0. URL: http://apache-server.com/tutorials/ATusing-htaccess.html (26 Jan 2002)
- "Sample Password Change Script." Article ID: W13310, Filename: Sample Change Password Script.txt, File Created: 1999:05:05:09:38:43 Last Updated: 2001:04:18:09:50:04. URL: Search for "W13310" at http://techsupt.windowware.com/webcgi/webbatch.exe?techsupt/techsupt.web (26 Jan 2002)
- "World Wide Web Consortium (W3C) FAQ." Version 3.1.1, Sept. 12, 2001. URL: http://www.w3.org/Security/Faq/www-security-faq.html (26 Jan 2002)
- "Totals for Top Servers Across All Domains." Dec. 2001. URL:http://www.netcraft.com/survey (26 Jan 2002)
- Walker, John. "Autodesk, Inc". Fourth Edition, 1994. URL: http://www.fourmilab.ch/autofile (26 Jan 2002)
- SSH Communications Security. URL: http://www.ssh.org (26 Jan 2002)
- "Srsh Simplified Restricted Shell 0.1.3." SecurityFocus™, by fygrave@scorpions.net. URL: http://www.securityfocus.com/tools/590 (26 Jan 2002)
- "Apache suEXEC Support." Apache HTTP Server Version 2.0. URL: http://httpd.apache.org/docs-2.0/suexec.html (26 Jan 2002)
- Perl™. URL: http://www.perl.org (26 Jan 2002)
- Rahe, Bill, and others. "gen_password" Jan. 1998. URL: http://www.lanl.gov/asci/DCE/Presentations/Workshop-Jan98/dce_accounts.ppt (26 Jan 2002)
- IDEA, International Data Encryption Algorithm. URL: ftp://ftp.cert.dfn.de/pub/tools/crypt/idea (26 Jan 2002)
- "The Expect Home Page" URL: http://expect.nist.gov (26 Jan 2002)
- Winbatch. URL: http://www.winbatch.com (26 Jan 2002)
- Hansknecht, Deborah. Sandia National Laboratories. URL: http://www.sandia.gov (26 Jan 2002)
- Quinn, Mickey G. "A Turtle On A Fence Post." If you see a turtle sitting on a fence post, just remember he had some help getting there. Jan 10,1995. URL: http://members.aol.com/mgquinn/turtle.html (26 Jan 2002)

© SANS Institute 2009, As part of the Information Security Reading Room Author retains full rights.

Last Updated: May 25th, 2013

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Malaysia @ MCMC 2013	Cyberjaya, MY	Jun 03, 2013 - Jun 08, 2013	Live Event
SANS Pen Test Berlin 2013	Berlin, DE	Jun 03, 2013 - Jun 08, 2013	Live Event
Industial Control Systems Security Training - Houston	Houston, TXUS	Jun 10, 2013 - Jun 15, 2013	Live Event
Security Impact of IPv6 Summit 2013	Washington, DCUS	Jun 14, 2013 - Jun 16, 2013	Live Event
SANSFIRE 2013	Washington, DCUS	Jun 14, 2013 - Jun 22, 2013	Live Event
SANS Canberra 2013	Canberra, AU	Jul 01, 2013 - Jul 13, 2013	Live Event
Digital Forensics & Incident Response Summit 2013	Austin, TXUS	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS London Summer 2013	London, GB	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS Rocky Mountain 2013	Denver, COUS	Jul 14, 2013 - Jul 20, 2013	Live Event
SANS Mumbai 2013	Mumbai, IN	Jul 22, 2013 - Jul 27, 2013	Live Event
SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification	Washington, DCUS	Jul 22, 2013 - Jul 26, 2013	Live Event
SANS San Francisco 2013	San Francisco, CAUS	Jul 29, 2013 - Aug 03, 2013	Live Event
SANS SEC 560: Network Penetration Testing @ Bangalore 2013	Bangalore, IN	Aug 05, 2013 - Aug 10, 2013	Live Event
SANS Boston 2013	Boston, MAUS	Aug 05, 2013 - Aug 10, 2013	Live Event
Critical Security Controls Summit	Washington, DCUS	Aug 12, 2013 - Aug 18, 2013	Live Event
Industrial Control Systems Security Training - DC	Washington, DCUS	Aug 12, 2013 - Aug 16, 2013	Live Event
SANS Thailand 2013	Bangkok, TH	Aug 19, 2013 - Aug 31, 2013	Live Event
SANS Virginia Beach 2013	Virginia Beach, VAUS	Aug 19, 2013 - Aug 30, 2013	Live Event
Mobile Device Security Summit 2013	OnlineCAUS	May 30, 2013 - Jun 06, 2013	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced