SANS Institute
InfoSec Reading Room

The Password Web Page

It is important to pick good passwords and change them often. This paper addresses the benefits and merits of the password web page.

The Password Web Page
Curt Kuper
January 12, 2002

Introduction

The Password Web Page is used to implement password selection. It is important to give computer users an easy way to choose and set good passwords. As stated by Garfinkel & Spafford in their book Practical UNIX & Internet Security [2], "making sure that users pick good passwords is one of the most important parts of running a secure computer system". Passwords play a key role in keeping the system secure.

The password web page helps keep systems secure by giving users an easy way to pick good computer generated passwords. The web page can be designed to run with any command line password generator. We will look on the Internet for password generators and research ways to safely encrypt and store the passwords. Once the user selects a password, it is encrypted and stored until a root level process can enter it onto the system. It is vital to pick a safe encryption algorithm to temporarily store the password.

Click here to see what the password web page looks like. The web page is created by a CGI program [10] and a password generator. The source code for the scripts is not included with this paper. These web pages can be designed by following the steps in the O'Reilly book CGI Programming on the World Wide Web by Shishir Gundavaram [3].

Background

The Password Web Page replaces a process where passwords were assigned and distributed to users. The passwords were generated and printed on a form for each user account. This form went into an envelope, was sealed and then distributed to each user. A lot of time and effort was involved in this process. The password web page helped us to automate the process and save some paper. It also saved the administrators time tracking down each person to hand them their envelope.

Under the old method, users did not choose their own passwords. They often received a password that was difficult to remember. This meant they would need to write down their password and refer to it often, especially in the first few days. Some users would comment on how their password would expire when they finally had it memorized. Also, if they liked the one they had before, they could pretty much guarantee that wouldn't be the case for the next password. It was a tough job handing out new passwords, you sure wouldn't make too many friends that day.

When the password is written down, you run the risk of it being seen by other people that you work with. If your password is seen by someone else, you need to ask the password administrator to assign you another one, which, knowing our password administrator,

your new password would definitely be ten times worse than the one you had before. So it is very important to keep your password private.

I remember one user (at a previous work location), who had a note on his monitor with a word written in a foreign language. Another system administrator that I worked with was able to read the note and asked the user if that was his password. Needless to say, it's not a good idea to write your password down and put it on your monitor or under your keyboard, even if it's written in another language.

Password Policy

Many organizations have problems due to weak passwords. Often on systems with no password policy, users tend to pick simple passwords and keep them in place for several years. I've heard computer security officials comment, "a chain is only as strong as it's weakest link". Even if the vulnerable system does not contain data that needs to be protected, it may be used as a staging area to attack other systems.

The password web page can help implement your password policy. For example, let's look at some items in a sample password policy:

1. Passwords must be eight characters long.
2. Passwords must contain an uppercase letter, at least one number, and/or one or more special characters.
3. Passwords must be computer generated.
4. New passwords must be unique and can not match previous passwords.

5. Passwords expire after 6 months.
6. The account is locked after 3 failed login attempts.
7. If passwords are written down, keep them in a secure location.
8. Do not store unencrypted passwords on the system.
9. If your password is compromised, change it right away and notify the computer security officer.
10. Do not use programs like telnet and ftp that allow the password to show up in clear text over the network.

Your choice of which password generator to use will be important in items 1-4. Password generators usually produce passwords that are harder to crack than passwords chosen by a user. They also give you choices on the length and the number of uppercase and special characters to enter in the password. Other password generators might focus on giving you "pronounceable" passwords that are easier to remember. If the password is easy to remember then the user is less likely to write it down, a strong advantage for using "pronounceable" passwords.

Other important items mentioned in lines 5-10, can be added to an introduction to the password web page. This page can serve to notify the users of their responsibilities to

keep the password secure. The introduction page can also provide contact information on who the computer security officer is and on who to go to if they have problems with the password web page.

Our passwords expire every six months. The password web page can be implemented in two ways. One way would be to notify all the users every 6 months (on June 1st and December 1st for example) that they need to go to the password web page and choose a new password. Another option would be to continuously monitor how old the users passwords are and send each one a notice when they are close to the 6 month expiration date.

We can determine the expiration date on each user account from the number in the third field of the /etc/shadow file. That entry is a count of the number of days from January 1, 1970 (the Unix epoch) to the date when the password was last changed. To determine how old the users' password is, I compute the current number of days from the Unix epoch and subtract the entry from the shadow file. I use the time function in a Perl script as follows:

```
# compute the number of days from Jan 1, 1970
$days = int( time() / (60 * 60 * 24) );
```

Design of the Password Web Page

The password web page was designed to give users an easy way to select their own password. The passwords that we assign are difficult to remember. Take a look at this demo page to see how bad the passwords are. Can you imagine being assigned the password "3icxaYkw" for six months? I felt it was important to give the users a way to pick their own password. There should be at least one out of the 30 passwords on the web page that the user would like. If not, there's a button that will generate another 30 passwords.

The web page is located on an SGI server running the Apache web server software. The mod_ssl package was added to encrypt the connection to the password web page. The software can be obtained from http://www.apache.org/ and http://www.modssl.org/. Information on how to install and configure Apache and mod_ssl can also be obtained from those sites. I won't discuss the configuration here except to say that it's best not to run the Apache daemon as root.

Let's take another look at the password web page and see how it's setup. Click here to see what the password web page looks like. Before the user sees this "password change form", the user is directed to an introductory page. After they've read the notices and reminders on the password policies, they click on a link to bring up the "password change form". However, before they get the form, a login box appears since the server is running mod_ssl and we want the password selection to be encrypted.

After the user logs in, the CGI program can obtain the "username" which they logged in as, and the user gets a personalized "password change form". Logs for the connection are also kept on the server. This allows the administrator to verify the connection if any problems occur. We had an instance once where a user selected a password but mistyped it and didn't notice the error message after he hit the submit button. By checking the logs, I was able to let him know what caused the problem.

The output of the password generator is used to build a table on the web page. It is important to choose a password generator that is appropriate for your site. I will discuss three password generators in the next section.

When the user hits the "submit password" button, the password entries are checked to make sure they match. The script also checks to make sure the password entries were taken from the current table. Then another page appears and either an error message is given or the user is notified that the password entry has been accepted. The valid password is encrypted and stored on the server.

When passing parameters in the CGI programs, to be more secure use "POST" instead of "GET". Other scripts run on the server to activate the password and copy them to the Unix desktops. These scripts have saved the administrators a lot of time. They no longer have to distribute the passwords to each user.

Password Generators

The password web page is built around a password generator. It can be designed to run with any command line password generator that is appropriate for your site and meets the requirements of your password policy. Searching on the Internet, I found several interesting programs. The three password generators that I would recommend using are: Tom Van Vleck's gpw, apg from Adel Mirzazhanov, and passwdGen by Denis Lemire. Let's take a closer look at what these programs offer.

gpw Password Generator [7]

One of the first password generators I found that generates pronounceable passwords is "gpw". The passwords are pronounceable so they are fairly easy to remember. The program was written by Tom Van Vleck and is available from his web site, http://www.multicians.org/thvv/tvvtools.html#gpw . I first noticed gpw a few years ago at sunfreeware.com in the SPARC/Solaris 2.6 section. An example of passwords you receive from gpw are: "heatchar, roureven, nusetteu, brimicen, and rendoodi". The pronounceable passwords are nice but one problem with gpw is that it does not offer an option to add uppercase and special characters.

Automated Password Generator (APG) [8]

APG is another password generator that generates pronounceable passwords. According to NIST (National Institute of Standards and Technology) publication FIPS (Federal Information Processing Standards) 181, APG is a standard password generator to be used by Federal organizations to authorize access to system resources. APG was written by Adel I. Mirzazhanov and is available from the APG home page at http://www.adel.nursat.kz/apg/ . Here's an example run of apg version 2.0.0 using the default options:

```
% apg
CAN NOT USE /dev/random TO GENERATE RANDOM SEED
USEING LOCAL TIME FOR SEED GENERATION !!!
coopho (coop-ho)
Bangyag= (Bang-yag-EQUAL_SIGN)
mipal5 (mip-al-FIVE)
yillyee (yill-yee)
DudQui (Dud-Qui)
myShteo (my-Shteo)
```

Special symbols can be added with the "-M" option. APG also has the ability to check the generated passwords and make sure they don't appear in a dictionary file. You can add usernames and other words to that file to make sure users don't receive those passwords. Dictionaries that come with password assessment programs can be used.

passwdGen - Random Password Generator [9]

The final password generator that I'll review in this paper is passwdGen, version 2.2. It was written entirely in C++ by Denis Lemire and is available from http://www.members.home.com/denisl/passwdgen . Similar to the previously mentioned password generators, paswdGen has the ability to generate pronounceable passwords. Also a unique feature that I haven't seen in other password generators, passwdGen can generate passwords of characters from entirely left keyboard, right keyboard, or alternating keys. Here's some examples from passwdgen:

```
% passwdgen -ap --length=8
Your password is: xifebaga
% passwdgen -Aap --length=8
Your password is: biCOKaju
% passwdgen -Aap --length=8
Your password is: xaxADutO
```

Encrypting the Password

The password web page allows a user to select a password. Once the user hits the "Submit Password" button, the password needs to be temporarily stored on the server.

The web server process does not run as root, so it can not set the password on the server. The password needs to be encrypted and stored in a protected area.

I researched encryption algorithms on the Internet and found a couple that would meet our needs to safely store the selected passwords. One of the encryption algorithms is IDEA, the International Data Encryption Algorithm, and the other is GnuPG, the GNU Privacy Guard encryption tool. Unix also has a command called "crypt", but don't use it for storing passwords. I'll explain why you don't want to use crypt in the next section, IDEA and GnuPG are much better choices.

The Unix crypt Command [4]

The crypt command is found on most Unix systems. It is used to encrypt and decrypt the contents of a file. This is good to do, but the problem with crypt is that it is very easy to break. That fact is even mentioned in the man page for crypt(1): "crypt implements a one-rotor machine designed along the lines of the German Enigma, but with a 256-element rotor. Methods of attack on such machines are widely known, thus crypt provides minimal security. ... The choice of keys and key security are the most vulnerable aspect of crypt." [4]

Even though crypt is not considered to be safe to use, it is better than leaving the file unencrypted. There are also ways to improve the security of crypt mentioned in the O'Reilly book, Practical UNIX & Internet Security [2]. One way would be to simply compress the file before encrypting it. Another option to improve security would be to run crypt on the file several times and use a different key each time. Also, don't confuse the crypt command with the crypt() library function. The crypt() function is used to encrypt user passwords in the /etc/shadow file and is considered to be safe.

Here is an example on how to use crypt to encrypt the contents of "input_file" and send the output to "output_file":

```
% crypt  < input_file  > output_file
Enter key:
```

Then to decrypt the file:

```
% crypt  < output_file
Enter key:
...
[the decrypted contents of the file are sent to standard out]
```

International Data Encryption Algorithm (IDEA) [5]

IDEA is a single-key encryption algorithm that is considered to be highly secure. It is the block cipher algorithm used in PGP (Pretty Good Privacy) for secure communications. IDEA is more secure than DES (Data Encryption Standard) since it uses an 128 bit encryption key compared to 56 bits for DES. It also runs much faster than the DES and RSA (Rivest, Shamir, and Adleman) software. [2]

The IDEA software is available from ftp.isi.ee.ethz.ch/pub/simpl/idea.V1.2.tar.Z . A DOS executable, idea.exe, is also available to run with MS-Windows PC's. There is no charge for noncommercial use. IDEA is protected by an international patent but on the plus side, there are no export restrictions like there are for DES.

Here is an example on how to use idea to encrypt the contents of "input_file" and send the output to "output_file":
% idea -e -k key input_file output_file

Then to decrypt the file:
% idea -dk key output_file
..
[the decrypted contents of the file are sent to standard out]

GNU Privacy Guard (GnuPG) [6]

GnuPG is a tool for secure communication and is a replacement for PGP (Pretty Good Privacy). However, it can also be used to just encrypt data. GnuPG is free software and it does not use the patented IDEA algorithm. The GnuPG software and documentation is available from http://www.gnupg.org/gnupg.html. Precompiled binaries are available for MS-Windows.

Here is an example on how to use gpg to encrypt the contents of "input_file" and send the output to "output_file":
% gpg --output output_file --symmetric input_file
Enter passphrase:
Repeat passphrase:

Then to decrypt the file:
% gpg --decrypt output_file
Enter passphrase:
..
[the decrypted contents of the file are sent to standard out]

Conclusion

It is important to pick good passwords and change them often. The password web page helps us implement this by giving the users a list of good computer generated passwords to choose from. The users can search through the list of passwords and also generate a new list. So giving the users a choice of passwords should make it easier for them to find one that they will be able to remember.

References:

[1]. SecurityFocus, http://www.securityfocus.com/
"The Leading Provider of Security Intelligence Services for Business"

[2]. Simson Garfinkel & Gene Spafford. Practical UNIX & Internet Security, Second Edition, April 1996.
Published by O'Reilly & Associates, Inc., ISBN 1-56592-148-8, 1004 pages.

[3]. Shishir Gundavaram. CGI Programming on the World Wide Web, First Edition, March 1996.
Published by O'Reilly & Associates, Inc., ISBN 1-56592-168-2, 436 pages.

[4]. Unix crypt(1) command
SunOS 5.8 man page. Last change: 14 May 1997

[5]. International Data Encryption Algorithm (IDEA).
Developed by Dr. Xuejia Lai and Dr. J. L. Massey at the Swiss Federal Institute of Technology.
International patent WO 91/18459, author - Richard De Moliner (demoliner@isi.ee.ethz.ch).
Signal and Information Processing Laboratory, Swiss Federal Institute of Technology, CH-8092 Zurich, Switzerland
Obtained from ftp.isi.ee.ethz.ch/pub/simpl/idea.V1.2.tar.Z

[6]. The GNU Privacy Guard (GnuPG).
Software and documentation available from http://www.gnupg.org/
Created by the GnuPG team: Matthew Skala, Michael Roth, Niklas Hernaeus, Rémi Guyomarch and Werner Koch. Gael Queri, Gregory Steuck, Janusz A. Urbanowicz, Marco d'Itri, Thiago Jung Bauermann, Urko Lusa and Walter Koch did the official translations. Mike Ashley is working on the GNU Privacy Handbook.

[7]. gpw password generator
Written by Tom Van Vleck, http://www.multicians.org/thvv/tvvtools.html#gpw
Also found on http://sunfreeware.com/ in the SPARC/Solaris 2.6 section.

[8]. Automated Password Generator (APG) v2.0.0
Written by Adel I. Mirzazhanov
APG Home page: http://www.adel.nursat.kz/apg/
The Software is also available from http://www.securityfocus.com
See NIST publication FIPS 181, ``Automated Password Generator", 5 Oct. 1993, http://www.itl.nist.gov/fipspubs/fip181.htm

[9]. passwdGen - Random Password Generator 2.2
Written by Denis Lemire, denis@lemire.com, http://www.members.home.com/denisl/passwdgen

[10]. Password Web Page CGI Scripts
Written by Mark Holbrook.

Last Updated: August 16th, 2017

Upcoming SANS Training

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

SANS Chicago 2017	Chicago, ILUS	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VAUS	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Adelaide 2017	OnlineAU	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced