User Authentication HOWTO

Peter Hernberg

2000/05/02

Explains how user and group information is stored and how users are authenticated on a Linux system (PAM), and how to secure you system's user authentication.

Table of Contents

1. Introduction......1
1.1. How this document came to be......1
1.2. New versions......1
1.3. Feedback......1
1.4. Version History......1
1.5. Copyrights and Trademarks......1
1.6. Acknowledgements and Thanks......2
1.7. Assumptions about the reader......2

2. How User Information is Stored on Your System......3
2.1. /etc/passwd......3
2.2. Shadow passwords......3
2.3. /etc/group and /etc/gshadow......3
2.4. MD5 encrypted passwords......4
2.5. Sifting through the mess......4

3. PAM (Pluggable Authentication Modules)......5
3.1. Why......5
3.2. What......5
3.2.1. Distributions that support pam......5
3.2.2. Installing PAM......6
3.3. How......6
3.3.1. PAM configuration files......6
3.3.2. A little something......6
3.3.3. Configuration syntax......7
3.3.4. pam.conf configuration......8
3.4. Getting more information......9

4. Securing User Authentication......10
4.1. A strong /etc/pam.d/other......10
4.1.1. A paranoid configuration......10
4.1.2. A kinder configuration......10
4.1.3. Choosing a /etc/pam.d/other......11
4.2. Disabling logins for user with null passwords......11
4.3. Disable unused services......11
4.4. Password−cracking tools......11
4.5. Shadow and MD5 passwords......12

5. Tying it all together......13
5.1. Apache + mod_auth_pam......13
5.1. Our example......13
5.2. Installing mod_auth_pam......13
5.3. Configuring PAM......13
5.3.1. Deciding how to configure PAM......14
5.4. Configuring Apache......14
5.5. Testing our setup......15

Table of Contents

6. Resources...16
- 6.1. PAM...16
- 6.2. General Security...16
- 6.3. Offline Documentation...16

7. Conclusion...17

1. Introduction

1.1. How this document came to be

When trying to add a number of (mostly unnecessary :) network services to my existing home network, I kept running into the problem of authentication, so I decided to figure out how authentication works on linux systems, write a HOWTO, and call it my senior project. I hope this document helps you understand this often−forgotten, but very important, aspect of system administration.

1.2. New versions

When I get my domain up running properly, you'll be able to find the newest version of this document there. Until then, http://www.linuxdoc.org/ will have to suffice.

1.3. Feedback

Comments, corrections, suggestions, flames, and flying saucer sightings can be sent to petehern@yahoo.com.

1.4. Version History

v0.1 (May 13, 2000) first version (not released).

v0.3 (May 14, 2000) revised (not released).

v0.5 (May 15, 2000) added section on securing pam, added resources section (not released).

v0.7 (May 15, 2000) revised; ready for release.

1.5. Copyrights and Trademarks

1.6. Acknowledgements and Thanks

Thanks to my family for putting up with me for 18 years. Thanks to the Debian folks for making such a sweet distro for me to play with. Thanks to CGR for paying me to be a geek. Thanks to Sandy Harris for his helpful suggestions. Finally, I'd like thank the makers of ramen noodles, because I don't know how I'd live without them.

1.7. Assumptions about the reader

For the purpose of this document, it is assumed that the reader is comfortably with executing commands at the command line and editing text configuration files.

2. How User Information is Stored on Your System

2.1. `/etc/passwd`

On almost all linux distributions (and commercial *nixes as well), user information is stored in `/etc/passwd`, a text file which contains the user's login, their encrypted password, a unique numerical user id (called the uid), a numerical group id (called the gid), an optional comment field (usually containing such items as their real name, phone number, etc.), their home directory, and their preferred shell. A typical entry in `/etc/passwd` looks something like this:

```
pete:K3xcO1Qnx8LFN:1000:1000:Peter Hernberg,,,1-800-FOOBAR:/home/pete:/bin/bash
```

As you can see, it's pretty straight−forward. Each entry contains the six fields I described above, with each field separated by a colon. If this were as complex as user authentication got, there would be no need for this HOWTO.

2.2. Shadow passwords

Looking at your `/etc/passwd`, it's likely that you actually saw something like this:

```
pete:x:1000:1000:Peter Hernberg,,,1-800-FOOBAR:/home/pete:/bin/bash
```

Where did the encrypted password go? Before I tell you where it went, a bit explanation is required.

The `/etc/passwd` file, which contains information about all users, including their encrypted password, is readable by all users, making it possible for any user to get the encrypted password of everyone on the system. Though the passwords are encrypted, password−cracking programs are widely available. To combat this growing security threat, shadow passwords were developed.

When a system has shadow passwords enabled, the password field in `/etc/passwd` is replaced by an "x" and the user's real encrypted password is stored in `/etc/shadow`. Because `/etc/shadow` is only readable by the root user, malicious users cannot crack their fellow users' passwords. Each entry in `/etc/shadow` contains the user's login, their encrypted password, and a number of fields relating to password expiration. A typical entry looks like this:

```
pete:/3GJllg1o4152:11009:0:99999:7:::
```

2.3. `/etc/group` and `/etc/gshadow`

Group information is stored in `/etc/group`. The format is similar to that of `/etc/passwd`, with the entries containing fields for the group name, password, numerical id (gid), and a comma−separated list of group members. An entry in `/etc/group` looks like this:

```
pasta:x:103:spagetti,fettucini,linguine,vermicelli
```

As you can see from the "x" in the password field, group passwords can be shadowed as well. Although groups almost never have their own passwords, it is worth noting that shadowed group password information is stored in `/etc/gshadow`.

2.4. MD5 encrypted passwords

Traditionally, unix passwords were encrypted with the standard crypt() function. (For more information on the crypt() function, see the crypt(3) manpage.) As computers grew faster, passwords encrypted with this function became easier to crack. As the internet emerged, tools for distributed the task of password−cracking across multiple hosts became available. Many newer distributions ship with the option of encrypting passwords with the stronger MD5 hash algorithm. (For more information on the MD5 hash algorithm, consult RFC 1321.) While MD5 passwords will not eliminate the threat of password cracking, they will make cracking your passwords much more difficult.

2.5. Sifting through the mess

As you can see, there are a number of different ways user authentication information can be stored on your system (shadow passwords without MD5 encryption, `/etc/passwd` passwords with MD5 encryption, etc.). How do programs like login and su know how to verify your password? Worse yet, what if you wanted to change the way passwords are stored on your system? How will programs that need your password know that passwords are stored differently? PAM is the answer.

3. PAM (Pluggable Authentication Modules)

Pluggable authentication modules are at the core of user authentication in any modern linux distribution.

3.1. Why

Back in the good old days of linux, if a program, such as su, passwd, login, or xlock, needed to authenticate a user, it would simply read the necessary information from `/etc/passwd`. If it needed to change the users' password, it would simply edit `/etc/passwd`. This simple but clumsy method presented numerous problems for system administrators and application developers. As MD5 and shadow passwords became increasingly popular, each program requiring user authentication had to know how to get the proper information when dealing with a number of different schemes. If you wanted to change your user authentication scheme, all these programs had to be recompiled. PAM eliminates this mess by enabling programs to transparently authenticate users, regardless of how user information is stored.

3.2. What

Quoting from the Linux−PAM System Administrator's Guide: "It is the purpose of the Linux−PAM project to separate the development of privilege granting software from the development of secure and appropriate authentication schemes. This is accomplished by providing a library of functions that an application may use to request that a user be authenticated." With PAM, it doesn't matter whether your password is stored in /etc/passwd or on a server in Hong Kong. When a program needs to authenticate a user, PAM provides a library containing the functions for the proper authentication scheme. Because this library is loaded dynamically, changing authentication schemes can be done by simply editing a configuration file.

Flexibility is one of PAM's greatest strengths. PAM can be configured to deny certain programs the right to authenticate users, to only allow certain users to be authenticated, to warn when certain programs attempt to authenticate, or even to deprive all users of login privileges. PAM's modular design gives you complete control over how users are authenticated.

3.2.1. Distributions that support pam.

Nearly all popular distributions have supported PAM for some time. Here's an incomplete list of distributions that support PAM:

- Redhat since version 5.0
- Mandrake since 5.2
- Debian since version 2.1 (partial support in 2.1 −− complete support in 2.2)
- Caldera since version 1.3
- Turbolinux since version 3.6
- SuSE since version 6.2

This list is certainly incomplete and possibly inaccurate. I'd appreciate it if you sent any corrections or additions to this list to `<petehern@yahoo.com>`.

3.2.2. Installing PAM

Installing PAM from scratch is long process, beyond the scope of this HOWTO. If PAM isn't installed on your system, you're probably running such an old version of your distribution that there are many other reasons to upgrade. If you really want to do it yourself, then you're certainly not the sort of person who needs any help from me. For all these reasons, I'm going to assume that you already have PAM installed.

3.3. How

Enough talk, let's dig in.

3.3.1. PAM configuration files

PAM configuration files are stored in the `/etc/pam.d/` directory. (If you don't have `/etc/pam.d/` directory, don't worry, I'll cover that in the next section) Let's go over there and take look.

```
~$ cd /etc/pam.d
  /etc/pam.d/$ ls
  chfn  chsh    login   other   passwd  su      xlock
  /etc/pam.d/$
```

Your system may have a few more or a few less files in this directory, depending on what's installed on your system. Whatever the details, you probably saw a file for each of the programs on your system that authenticate users. As you probably already guessed, each file contains the PAM authentication configuration for the program it's named after (except for the `other` file, which we'll talk about in a little bit). Let's take a look the PAM configuration file for passwd (I've condensed the file for the sake of simplicity):

```
/etc/pam.d/$ cat login
  # PAM configuration for login
  auth       requisite  pam_securetty.so
  auth       required   pam_nologin.so
  auth       required   pam_env.so
  auth       required   pam_unix.so nulok
  account    required   pam_unix.so
  session    required   pam_unix.so
  session    optional   pam_lastlog.so
  password   required   pam_unix.so nullok obscure min=4 max=8
```

Before dig into this file, I must mention a little something.

3.3.2. A little something

A small percentage are probably thinking, "Oh no! I don't have a /etc/pam.d directory! Your list of distributions says that my distribution includes PAM, but I can't find that directory. Without PAM, my life is empty and meaningless! What can I do?" Don't worry, all is not lost. If you know that your distribution includes PAM, but you have no `/etc/pam.d/` directory, then your PAM configuration is stored in `/etc/pam.conf`. Rather than being spread across several files, all your PAM configuration is stored in a single file. This adds a little twist to PAM configuration, but the proper adjustments are pointed out in section 3.3.4.

3.3.3. Configuration syntax

PAM configuration files have the following syntax:

```
type  control  module-path  module-arguments
```

Using the login configuration file (see above) as an example let's take a look a the syntax for PAM configuration files:

PAM configuration tokens

type

The type token tells PAM what type of authentication is to be used for this module. Modules of the same type can be "stacked", requiring a user to meet multiple requirements to be authenticated. PAM recognizes four types:

account

- Determines whether the user is allowed to access the service, whether their passwords has expired, etc.

auth

Determines whether the user is who they claim to be, usually by a password, but perhaps by a more sophistcated means, such as biometrics.

password

Provides a mechanism for the user to change their authentication. Again, this usually their password.

session

Things that should be done before and/or after the user is authenticed. This might included things such as mounting/unmounting the user home directory, logging their login/logout, and restricting/unrestricting the services available to the user.

In the login config file, we see at least one entry for each type. Since this the program that allows user to login (hence the name :), it's understandable that it needs to access all of the different types of authentication.

control

The control token tells PAM what should be done in if authentication by this module fails. PAM recognizes four control types:

requisite

- Failure to authenticate via this module results in immediate denial of authentication.

required

- Failure also results in denial of authentication, although PAM will still call all the other modules listed for this service before denying authentication.

sufficient

If authentication by this module is successful, PAM will grant authentication, even if a previous required module failed.

optional

Whether this module succeeds or fails is only significant if it is the only module of its type for this service.

In the configuration file for login, we see nearly all of the different control types. Most of the required modules are pam_unix.so (the main authentication module), the single requisite module is pam_securetty.so (checks make sure the user is logging in on a secure console), and the only optional module is pam_lastlogin.so (the module that retrieves information on the user's most recent login).

module−path

The module−path tells PAM which module to use and (optionally) where to find it. Most configurations only contain the module's name, as is the case in our login configuration file. When this is the case, PAM looks for the modules in the default PAM module directory, normally `/usr/lib/security`. However, if your linux distribution conforms to the Linux Filesystem standard, PAM modules can be found in `/lib/security`.

module−arguments

The module−arguments are arguments to be passed to the module. Each module has its own arguments. For example, in our login configuration, the "nulok" ("null ok", argument being passed to pam_unix.so module, indicating the a blank ("null") password is acceptable ("ok").

3.3.4. `pam.conf` configuration

If your PAM configuration is stored in `/etc/pam.conf` rather than `/etc/pam.d/`, PAM configuration lines are a bit different. Rather than each service having its own configuration file, all configurations are stored in `/etc/pam.conf` with the service name as the first token in a configuration line. For example, the following line in `/etc/pam.d/login`:

```
auth       required   pam_unix.so nulok
```

would become the following line in `/etc/pam.conf`:

```
login       auth       required   pam_unix.so nulok
```

Except for this minor difference, all the rest of the PAM syntax applies.

3.4. Getting more information

For more information on configuring PAM and complete PAM module reference, consult the Linux−PAM System Administrator's Guide. This guide serves as a thorough and up−to−date reference on PAM configuration.

4. Securing User Authentication

Many linux distributions ship with user authentication that is not adequately secure. This section discusses some of the ways you make user authentication secure on your system. While doing these things will make your system more secure, do not be so naive as to think they make you invulnerable.

4.1. A strong `/etc/pam.d/other`

All of the files in `/etc/pam.d/` contain the configuration for a particular service. The notable exception to this rule is the `/etc/pam.d/other` file. This file contains the configuration for any services which do not have their own configuration file. For example, is the (imaginary) xyz service attempted authentication PAM would look for a `/etc/pam.d/xyz` file. Not finding one, authentication for xyz would be determined by the `/etc/pam.d/other` file. Since `/etc/pam.d/other` is the configuration to which PAM services fallback, it is important that it is secure. We will discuss two secure configurations of `/etc/pam.d/other`, one which is quite nearly paranoid and which is gentler.

4.1.1. A paranoid configuration

A paranoid configuration of `/etc/pam.d/other` is as follows:

```
auth        required       pam_deny.so
auth        required       pam_warn.so
account     required       pam_deny.so
account     required       pam_warn.so
password    required       pam_deny.so
password    required       pam_warn.so
session     required       pam_deny.so
session     required       pam_warn.so
```

With this configuration, whenever an unknown service attempts to access any of the four configuration types, PAM denies authentication (via the pam_deny.so module) and then logs a syslog warning (via the pam_warn.so module). Short of a bug in PAM, this configuration is brutally secure. The only problem with that brutality is it may cause problems if your accidentally delete the configuration of another service. If your `/etc/pam.d/login` was mistakenly deleted, no one would be able to login!

4.1.2. A kinder configuration

Here's configuration that isn't quite so mean:

```
auth        required       pam_unix.so
auth        required       pam_warn.so
account     required       pam_unix.so
account     required       pam_warn.so
password    required       pam_deny.so
password    required       pam_warn.so
session     required       pam_unix.so
session     required       pam_warn.so
```

This configuration will allow an unknown service to authenticate (via the pam_unix.so module), although it will not allow it to change the user's password. Although it allows authentication by unknown services, it logs a syslog warning whenever such a service attempts authentication.

4.1.3. Choosing a `/etc/pam.d/other`

I would strongly reccomend that you implement the first `/etc/pam.d/other` configuration unless you have a very good reason not to. It always a good idea to be 'secure by default'. If you ever do need to grant a new service authentication privileges, you can simply create a PAM configuration file for that service.

4.2. Disabling logins for user with null passwords

On most linux systems, there a number of "dummy" user accounts, used to assign privileges to certain system services like ftp, webservers, and mail gateways. Having these accounts allows your system to be more secure, because if these services are compromised, an attacker will only gain the limited privileges available to the dummy account, rather than the full privileges of a service running as root. However, allowing these dummy account login privileges is a security risk, as they usually have blank (null) passwords. The configuration option that enables null passwords is the "nullok" module−argument. You'll want remove this argument from any modules of 'auth' type for services that allow login. This is usually the login service, may also include services like rlogin and ssh. Hence, the following line in `/etc/pam.d/login`:

```
auth        required        pam_unix.so     nullok
```

should be changed to:

```
auth        required        pam_unix.so
```

4.3. Disable unused services

Looking at the files in `/etc/pam.d/`, you'll probably see configuration files for a number of programs you don't use and maybe even a few you've never heard of. Although allowing authentication to these services probably won't open any huge security holes, you're better off denying them authentication. The best way to disable PAM authentication for these programs is to rename these files. Not finding the file named after the service requesting authentication, PAM will fallback to the (hopefully) very secure `/etc/pam.d/other`. If you later find that you need one of these programs, you can simply rename the file to its original name and everything will work as it was intended.

4.4. Password−cracking tools

While password−cracking tools can be by attackers used to compromise a system, they can also be used by system administrators as proactive tool to ensure the strength of passwords on their system. The two most commonly used password−cracking tools are "crack" and "John the Ripper". Crack is probably included in your facorite distribution. John the Ripper can be obtained from http://www.false.com/security/john/index.html. Run the tools against your password database and you'll probably be surprised with what they come up with.

Additionally, there is a PAM module which utilizes the crack library to check the strength of a users password whenever it changed. When this module is installed, the user can only change their password to one which meets the minimum password strength.

4.5. Shadow and MD5 passwords

As was discussed in the first section of this document, Shadow and MD5 passwords can make your system more secure. During the installation procedure, most modern distributions will ask whether you want to install MD5 and/or Shadow passwords. Unless you have a good reason not to, you should enable these. The process of converting from non−shadowed/non−MD5 passwords is a complicated process, and is beyond the scope of this document. The Shadow Password HOWTO is outdated, but it might be of some help.

5. Tying it all together

In this section, I'll give a simple example which ought to help tie together what's in the previous section.

5.1. Apache + mod_auth_pam

As our example, we'll install and configure mod_auth_pam, an Apache module that allows you to use authenticate users of your webserver using PAM. For the purpose of this example, I'll assume you have apache installed. If it's not installed already you should be able find installation packages from your distributor.

5.1. Our example

Our goal will be to configure a restricted area of our webserver, a `family/` directory, to authenticate users via PAM. This directory contains private family information, and should only be accessible to members of the user group family.

5.2. Installing mod_auth_pam

First, you'll want to download mod_auth_pam from http://blank.pages.de/pam/mod_auth_pam/. The following commands will compile mod_auth_pam (you must be logged in as root):

```
~# tar xzf mod_auth_pam.tar.gz
   ~# cd mod_auth_pam-1.0a
   ~/mod_auth_pam-1.0a# make
   ~/mod_auth_pam-1.0a# make install
```

If you have any trouble installing the mod_auth_pam module, make sure you've installed your distributions apache−dev package. After you've installed mod_auth_pam, you'll need to restart apache. Apache can usually by restarted by typing the following command (again, you must be root):

```
~# /etc/init.d/apache restart
```

5.3. Configuring PAM

PAM configuration for Apache is stored in `/etc/pam.d/httpd`. The default configuration (which was installed when you installed mod_auth_pam) is secure, but it uses a module (`pam_pwdb.so`) which may not be available on many systems. (Besides, configuring it from scratch will be fun!) So delete the `/etc/pam.d/httpd` file, and let's start fresh.

5.3.1. Deciding how to configure PAM

If we're going to configure how PAM deals with Apache's authentication requests, we need to figure out exactly what we need PAM to check for. First, we want PAM to make sure the user's password matches their password in the standard unix password database. This sounds like the 'auth' type and the `pam_unix.so` module. We'll want the module's control type to be set to 'required', so authentication will fail without a correct password. Here's what the first line of our `/etc/pam.d/httpd` looks like:

```
auth        required        pam_unix.so
```

Secondly, we must make sure that the users account is valid (i.e. their password has not expired or any such nastiness). This is the 'account' type and is also provided by the `pam_unix.so` module. Again, we'll set this module's control type to 'required'. After adding this line, our `/etc/pam.d/httpd` configuration looks like this:

```
auth        required        pam_unix.so
account     required        pam_unix.so
```

It's not terribly sophisticated, but it does the job. It ought to be a good start for learning how to configure PAM services.

5.4. Configuring Apache

Now that PAM is configured to authenticate apache's requests, we'll configure apache to properly utilize PAM authentication to restrict access to the `family/` directory. To do so, add the following lines to your `httpd.conf` (usually stored in `/etc/apache/` or `/etc/httpd`):

```
<Directory /var/www/family>
AuthPAM_Enabled on
AllowOverride None
AuthName "Family Secrets"
AuthType "basic"
require group family
</Directory>
```

You may need to replace `/var/www/` with the default location of web documents, which is often `/home/httpd/`. Wherever that is, you'll need to create the `family` directory.

Before we test our setup, I'll to take a moment to explain the Apache configuration you just entered. The <Directory> directive is used to encapsulate configuration data for this directory. Inside this directive, we've enabled PAM authentication ("AuthPAM_enabled on"), turned off any overriding of this configuration ("AllowOverride none"), named this authentication zone "Family Secrets" ("AuthName "Family Secrets""), set the http authentication (not the PAM authentication) type to the default ("AuthType "basic""), and required the user group family ("require group family").

5.5. Testing our setup

Now that we've got everything setup up properly, it's time to revel in our success. Fire up your favorite web browser and head over to http://your−domain/family/ (replacing your−domain with, well, your domain). You are now an uber−authenticator!

---

6. Resources

There are a number of resources, both online and offline, where you can more information about user authentication. If you know of any resources that ought to be added to this list, drop me a line at `<petehern@yahoo.com>`

6.1. PAM

- Linux−PAM System Administrator's Guide
- Linux−PAM Module Writer's Manual
- Linux−PAM Application Developer's Manual

6.2. General Security

- linuxsecurity.com
- securitywatch.com
- Security HOWTO
- Packetstorm

6.3. Offline Documentation

A lot of information can be gathered from your system's manual pages. The following are some manpages relating to user authentication. The number in parentheses refers to the manpage section. To view the passwd(5) manpage, you would enter **`man 5 passwd`**.

- passwd(5)
- crypt(3)
- pam.d(5)
- group(5)
- shadow(5)

7. Conclusion

I hope you found this HOWTO helpful. If you have any questions, comments, or suggestions, I'd love to hear from you. You can email me at `<petehern@yahoo.com>`.