How to delete user belonging to a group on FreeBSD - "the BSD (proper) way"

Author : admin

Some long time ago, I've created one user called **newuser**, on *my home FreeBSD router* and added him to be a member of **wheel** group.I've completely forgot about the users existing, just until yesterday when I saw the user still hanging around in my **wheel** group.For those unfamiliar with the **wheel** group on FreeBSD, wheel is the same like **root** group on Linux and some other *nices.

Before proceed with the reason fot this post **to show the proper way of adding and removing user to a group on BSD**, I will first explain a bit few things concerning BSD password files, where they are and why are they so many :)

On the first glimpse, people unfamiliar with BSD will be shocked / (confused) to find out there are 5 files, which has something to do to **password authentication**.

1. Some short explanation on /etc/passwd /etc/master.passwd, /etc/pwd.db, /etc/spwd.db, /etc/group and login.conf.db BSD auth and login files FreeBSD and rest of the BSD family has 5 files which deal with *username and password authentication, group ids, default shell configs* etc.:

The 5 ones are:

- /etc/passwd
- /etc/master.passwd
- /etc/group
- /etc/pwd.db
- /etc/spwd.db

/etc/passwd is readable by all the users on the system whether **/etc/master.passwd** is only readable by

root and **toor** administrative users. In that numbers members to *wheel* group have access for reading to all of the five.

Just like on Linux */etc/passwd* contains all kind of system existing users ... everything except the stored user passwords strings.
/etc/master.passwd is actually the BSD equivalent of Linux's **/etc/shadow** file. It stores md5 encrypted user passwords (by default) in a form of encrypted hashes. For tightened security one can, however choose to use a blowfish password hash encryption instead.

Since my *newuser* was a member to group, the user had read access to my /etc/master.passwd and hence this was a potential potential security hole on my system.
To close the whole I decided to remove *newuser*'s membership to *wheel* group.
Before I say how I actually did it. I will sawy few more words on *BSD systems authentication files structure*.

The file **/etc/master.passwd** is actually the BSD equivalent of Linux's **/etc/shadow**.
Besides */etc/password* and */etc/master.passwd*, on BSD there are also two other separate binary database files storing authentication user credentials:

```
freebsd# ls -l /etc/pwd.db /etc/spwd.db
-rw-r--r-- 1 root wheel 90112 Mar 13 23:56 /etc/pwd.db
-rw------- 1 root wheel 90112 Mar 13 23:56 /etc/spwd.db
```

In case if you're wondering what are this two *pwd.db files for:
/etc/pwd.db contains in database format /etc/passwd content
/etc/spwd.db contains in database format /etc/master.password
, **s**pwd.db stands for *(shadow) pwd.db*.
Near the end of the man page for *pwd_mkdb*, pwd.db is described as **"insecure password database file** and *spwd.db* as **secure password database file**.
The exact database type can be displayed with **file** command which is alawys helpful in (determining a file types).
I use **file** almost daily to check the (MIME) type of most of the "weird" file type extensions I have on my system. If not yet familiar with *file* cmd, be sure to try it on few various file extensions and see how it works.

```
freebsd# file /etc/pwd.db
/etc/pwd.db: Berkeley DB 1.85 (Hash, version 2, native byte-order)
freebsd# file /etc/spwd.db
/etc/spwd.db: Berkeley DB 1.85 (Hash, version 2, native byte-order)
```

You see, files are stored in format of Berkley DB Hash version 2.
The two files got updated every time with command **pwd_mkdb** whether a change in **/etc/master.passwd** occurs through use of lets say **pw** or **vipw**.
Btw, one common way to initiate changes to */etc/master.passwd* (lets say modify a user shell) is possible through **vipw** command.

vipw is a wrapper command that launch instance of *vi* editor over /etc/master.passwd, once changes are saved in the file, **pwd_mkdb** is run to regenerate the */etc/pwd.db* and */etc/spwd.db*. With this in mind **vipw** on BSD is the equivalent of manually editting */etc/shadow* with **vi /etc/shadow** on G / Linux.

Whether talking about user credentials and */etc/pwd.db* and */etc/spwd.db*, its worthy to mention there is one more **db** file - **/etc/login.conf.db**. */etc/login.conf.db* is red everytime a user logs in the system. It is is generated from the plain text */etc/login.conf*. Just in case if wondering why this *.db* files are used on FreeBSD at all, the reason is efficiency.
Reading binary database (structured data) as we all know is way faster than plain text file look ups
The performance advantage of the BSD's use of *.db* stored credentials is not so-"visible" in normal BSD systems with less than lets say 100 users.
Anyways on systems with few thousands of users that *login and logout frequently* the speed difference will surely be clear.

Manual generation of */etc/pwd.db* and */etc/spwd.db* or */etc/login.conf.db* is possible via **pwd_mkdb** and **cap_mkdb** commands.

After explaining shortly the basic auth files, I'll proceed with my specific case and will explain how I removed my *newuser* from membership in *wheel group*.

2. "BSD way" to remove or add existing user to member a group

The record for my user **newuser** in /etc/group, looked like so:

freebsd# grep -i newuser /etc/groupwheel:*:0:root,hipo,newuser

I was curious if **/etc/group** was possible to manually edit like on Linux with *vi* or *mcedit*.I thought this might be a problem since I thought the /etc/group info might be stored somewhere along in /etc/pwd.db or /etc/spwd.db. My hypothesis, however was wrong.
Straight use of *vim /etc/group* and deletion of the *newuser* record was enough to remove the user from wheel.

Anyways this is not a standard way and especially if it has to be scripted it is unnecessery hassel, hence below is the 'BSD way' via **pw**:

freebsd# pw groupmod wheel -d newuser

There is no output returned, therefore the command executed succesfully.

pw can be used for plenty of user management operations. Lets say I want to add back the *newuser* to be a member of wheel some time in the future, I could use:

freebsd# pw groupmod wheel -m newuser

To later check if newuser is succesfully removed from **/etc/group**:

```
freebsd# grep -i wheel /etc/group
wheel:*:0:root,hipo
```

Generally it is better, to stick to one way to do everything related to user and group management with **pw** and use it to show group permissions for wheel instead:

```
freebsd# pw group show wheel
wheel:*:0:root,hipo
```