User Administration

Managing passwords

Change current user password, prompt for current passwd - `passwd`
If you can sudo, run `sudo passwd <user>` to change a user password without prompt for current password, and with no security restrictions (min length, difficulty, etc)

Removing users

To remove a user, run `sudo userdel username`. To remove a user *and* their files within their `/home/username/` directory, run `sudo userdel -r username`

Adding users

> For a useful script to speed up this process when adding multiple users, skip to the end of this guide.

Run the following commands to create a new user on Linux -

These commands assume you are root, on a new host, so you do not need to prefix them with `sudo`, if you are not root you will need to run `sudo adduser <username>`, etc.

```
adduser username
Adding user `username' ...
Adding new group `username' (1000) ...
Adding new user `username' (1000) with group `username' ...
Creating home directory `/home/username' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for username
Enter the new value, or press ENTER for the default
        Full Name []: # You can leave all of this blank, or not
        Room Number []:# Your choice, really
        Work Phone []:
        Home Phone []:
```

```
        Other []:
Is the information correct? [Y/n] y
```

Configuring Sudo

Now, we need to configure the user for sudo access, so we set our preferred text editor and use `sudo -E` to preserve our user's environment settings while running commands as sudo.

```
# Set vim as our preferred editor
export EDITOR=/bin/vim && export VISUAL=/bin/vim
# Edit the sudoers file, preserving our current user's environment settings
sudo -E visudo
```

Find the section within `/etc/sudoers` called user privilege specification

```
# User privilege specification
root ALL=(ALL:ALL) ALL
```

Modify the file by adding the user to the section as it appears below, granting all permissions -

```
# User privilege specification
root ALL=(ALL:ALL) ALL
username ALL=(ALL:ALL) ALL
```

It's considered better practice to override the `/etc/sudoers` file by running `sudo visudo -f /etc/sudoers.d/mySudoers` - This command will allow us to store our changes in a file independent from the default sudoer configuration, and also complies with the idea that `/etc/sudoer` is a *sequential configuration,* which means the order in which settings are applied is crucial to how they are interpreted by our system.

If you feel your sudoer settings are being ignored, consider moving their location in `/etc/sudoers` to the end of the file, or use the command above to create a separate configuration, securing the default settings in the even that a mistake is made, we will still be able to authenticate using sudo. Save `/etc/sudoers` and quit, but note that you will need to logout and login again for your changes to take effect.

If you configured sudo access for your user, make sure you follow the next section to ensure they are added to the relevant `sudo` group

Configure Group Access

> Looking to check current group members? `sudo groupmems -l -g groupname`

> Want to add a single user to a single group? `sudo usermod -aG groupname username` will `-a` append the user to the given group. The `-G` option alone will remove the user from all groups other than the one provided.

Run `vigr` in the terminal and add your new username created to the sudo group, and any other groups you may want. This is the same thing as modifying the configuration file `/etc/group` with your preferred editor and saving it. (Docker is a common group that users will need added to - Don't run your containers as root by running `sudo docker`)

```
...
tape:x:26:
sudo:x:27:USERNAME,USERNAME2,USER3
audio:x:29:
docker:x:30:USERNAME,USER3
...
```

When saving `/etc/group`, you'll get some output warning you about consistency between a shadow configuration file. Go ahead and edit it to mirror your changes, and ignore the final warning about the `/etc/group` file consistency since we just came from modifying that file.

```
vigr
You have modified /etc/group.
You may need to modify /etc/gshadow for consistency.
Please use the command 'vigr -s' to do so.

vigr -s
You have modified /etc/gshadow.
You may need to modify /etc/group for consistency.
Please use the command 'vigr' to do so.
```

Securing User / Group IDs

You should change your user and group IDs from the default sequential values we can assume Linux has distributed for us. To do this, choose and valid ID and edit the following commands to suit your needs -

```
# Change user and group IDs
sudo usermod -u 1234 user
```

```
sudo groupmod -g 4321 usergroup

# Make sure you edit all the old permissions to reflect the above changes
# Use the old user and group IDs here
sudo find / -group 1000 -exec chgrp -h username {} \;
sudo find / -user 1000 -exec chown -h username {} \;
```

Not sure what UID and GID to choose? See the table below and choose a value that suits your needs - probably a value within an unused range. **UID and GID do not need to be the same** - This is only the case by default when adding a user via Linux Distributions such as Ubuntu, which is the one referenced / used in this guide. Feel free to specify unique values, and research more into sharing user groups for permissions in scenarios such as granting a list of employees or developers similar access.

> | UID/GID | Purpose | Defined By | Listed in |
> |---|---|---|---|
> | 0 | `root` user | Linux | `/etc/passwd` + `nss-systemd` |
> | 1 ... 4 | System users | Distributions | `/etc/passwd` |
> | 5 | `tty` group | `systemd` | `/etc/passwd` |
> | 6 ... 999 | System users | Distributions | `/etc/passwd` |
> | 1000 ... 60000 | Regular users | Distributions | `/etc/passwd` + LDAP/NIS/... |
> | 60001 ... 61183 | Unused | | |
> | 61184 ... 65519 | Dynamic service users | `systemd` | `nss-systemd` |
> | 65520 ... 65533 | Unused | | |
> | 65534 | `nobody` user | Linux | `/etc/passwd` + `nss-systemd` |
> | 65535 | 16bit `(uid_t) -1` | Linux | |
> | 65536 ... 524287 | Unused | | |
> | 524288 ... 1879048191 | Container UID ranges | `systemd` | `nss-mymachines` |
> | 1879048191 ... 2147483647 | Unused | | |

UID/GID	Purpose	Defined By	Listed in
2147483648 ... 4294967294	HIC SVNT LEONES		
4294967295	32bit `(uid_t) -1`	Linux	

Table Source - Systemd.io

You should validate all the configuration done to secure your server - for example, this could be validated by running the following commands to check UID / GID after setting them and logging into our user.

Check UID / GID

id -u username

id -g username

If you plan to stop here, be sure to login to your new user before making further changes to your system.

```
sudo su username
# Or
sudo -iu username
```

Bash Add User Script

Using the information on this page, we can create a simple bash script to handle this process for us. If you plan to add a fair amount of users to a system, automating at least the general portion of that process might be valuable to you. See the script below to automate up to this point in these instructions. Simply save it into `addusers.sh` for example, and run `sudo chmod a+x addusers.sh` followed by `sudo ./addusers.sh username 1005` where 1005 is the userID you wish to assign to your new user. Sudo is required here if you wish to assign sudo privileges to the new user.

> Want to call this from the commandline as any other command? Assuming you have the script marked as an executable placed within your `/opt/` directory, run `echo "export PATH=$PATH:/opt/" >> ~/.bash_aliases && source ~/.bashrc` You should now be able to run the script by its current name from any directory on the system - `adduser.sh` Feel free to rename it

```
#!/bin/bash
## Author: Shaun Reed | Contact: shaunrd0@gmail.com | URL: www.shaunreed.com ##
## A custom bash script for creating new linux users.                         ##
## Syntax: ./adduser.sh <username> <userID>                                   ##
```

```
################################################################################

if [ "$#" -ne 2 ]; then
  printf "Illegal number of parameters."
  printf "\nUsage: sudo ./adduser.sh <username> <groupid>"
  printf "\n\nAvailable groupd IDs:"
  printf "\n60001......61183 Unused | 65520..............65533  Unused"
  printf "\n65536.....524287 Unused | 1879048191.....2147483647  Unused\n"
  exit
fi

sudo adduser $1 --gecos "First Last,RoomNumber,WorkPhone,HomePhone" --disabled-password --uid
$2

printf "\nEnter 1 if $1 should have sudo privileges.  Any other value will continue and make
no changes\n"
read choice
if [ $choice -eq 1 ] ; then
printf "\nConfiguring sudo for $1...\n"
sudo usermod -G sudo $1
fi

printf "\nEnter 1 to set a password for $1, any other value will exit with no password set\n"
read choice

if [ $choice -eq 1 ] ; then
printf "\nChanging password for $1...\n"
sudo passwd $1
fi
```

Now after creating this user and following the prompts in the script above, all you'll need to do is configure the user-specific settings you wish to apply in your case.

Creating SSH Login Keys

If you don't plan to use this user for logging into another host, you shouldn't need to continue with ssh-keygen configuration. Simply add your public key to the authorized hosts file at `/home/username/.ssh/authorized_keys` so that you will be able to authenticate via SSH using your intended public key.

If you spun up a Droplet with DigitalOcean, they give you the ability to load the server with an approved public key. This means that there will already be an authorized_keys file created for us

with the same public key we used to login with our root user. Run the following commands from your user's home directory so you may authenticate with your public key.

```
sudo cp -r /root/.ssh .
sudo chown username .ssh/ && sudo chgrp username .ssh/
sudo chown username .ssh/authorized_keys && sudo chgrp username .ssh/authorized_keys
```

> Later, we will disable logging in with the root account - it's important to setup SSH access for other users beforehand!

SSH should never be authenticated using passwords alone, using public keys and ssh-keygen we can authenticate based on a a key we generate and distribute manually. This should be done with care, as a combination of sloppy authorized_keys files and lost / stolen keys can lead to a compromised web server!

```
sudo su username
cd
ssh-keygen -t ed25519
```

if you run the above command as sudo, it will create a key for root@host, not the user you are logged in as.
If you are getting privelege errors, you are not in your home directory. Create the key there first, then move it to your preferred location later. (usually /home/user/.ssh/)

general format for filename is user_ so -> username_ed25519
this will create a public and private key, the private key should(?) be backed up on an encrypted USB drive and removed from the server

Once the files are generated, they sit loose in the users home directory, let's clean them up..

```
mkdir .ssh
sudo mv username_ed25519* .ssh/


# Ensure permissions are set approprately  for .ssh/ and authorized_keys file
sudo chmod 700 ~/.ssh && sudo chmod 600 ~/.ssh/authorized_keys
```

Create an authorized_keys text file within the .ssh/ directory, and paste in the public key from within username_ed25519.pub - this is the file that will be checked for approved keys, the username_ed25519.pub is for your own records. Keep it there, delete it, put it on a usb, back it up.

Do not leave your private key on the server, should someone get this keyfile they can change your password and login as long as that key is on the approved list. These will unauthorized logins / password resets will not be viewed as a breach attempt, but as an approved login - no one will be alerted until its too late.

Using Putty with OpenSSH Keys

At some point when a password is used in key generation, ssh-keygen generates openssh private key which doesn't use cipher supported by puttygen.

ssh-keygen doesn't provide option to specify cipher name to encrypt the resulting openssh private key.

There is a workaround: remove the passphrase from the key before importing into puttygen.

Create a copy of the key to temporarily remove the password
`cp ~/.ssh/id_ed25519 ~/.ssh/id_ed25519-for-putty`

import the copied key, using the -p argument to specify a request to set a new password, and -f to specify the import keyfile.

```
ssh-keygen -p -f ~/.ssh/id_ed25519-for-putty
Enter old passphrase: <your passphrase>
Enter new passphrase (empty for no passphrase): <press Enter>
Enter same passphrase again: <press Enter>
```

using some command, view the text contents of the private key generated.

```
cat id_ed25519-for-putty
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZWQyNTUxOQ
AAACCGyjniPP1oVCXqkdCeCKFp+5+5cI7L79rP5RYHJ5Y6fQAAAJh3QGp1d0BqdQAAAAtzc2gtZWQy
NTUxOQAAACCGyjniPP1oVCXqkdCeCKFp+5+5cI7L79rP5RYHJ5Y6fQAAAEBJr8PzmuEN6qNyrY07Lr
LAgZRjo9efYETKqFbS2jVTQobKOeI8/WhUJeqR0J4IoWn7n7lwjsvv2s/lFgcnljp9AAAADmthcHBl
ckBrYXB1bnR1AQIDBAUGBw==
-----END OPENSSH PRIVATE KEY-----
```

copy this output from your ssh session to the machine running Putty

On the windows machine, create a .ssh directory in the users folder who wishes to SSH into the server (C:\Users\Shaun.ssh)

navigate inside the directory, and create a text file - paste the output from your private key into this file, file->saveAs In the dropdown 'save as file type', select 'All Files', be sure to end the

keyfile name with the .key extension -> username_ed25519.key click save.

Open puttygen, load convert->import keys.. select the text file we created in C:\Users\Shaun.ssh\ and set the passphrase from puttygen.

Don't forget to shred and remove ~/.ssh_id_ed25519-for-putty afterwards since it is not password protected.

The new password protected key will authorize the user based on the local password set in putty, using the remote PUBLIC key stored on the server.

Revision #32

Created Sat, Apr 6, 2019 5:08 AM by Shaun Reed

Updated Tue, May 19, 2020 9:04 PM by Shaun Reed