2023-01-07:  
protecting root

Cedar Rapids Area Homeschools’ 2023 Cyberdefense Team

# This Week’s Plan

We have a short meeting this week but will try to cover the following:

* Discuss ways to gain root-level access when connected directly to a Linux computer.
* Discuss ways to gain root-level access to a remote Linux computer.
* Discuss logging in as root interactively.
* Discuss safeguarding against logging in as root over remote protocols like **ssh**.
* Discuss safe configuration of the **sudo** command.
  + Create a new account for a trusted user named **alice** and another for an suspicious user named **mike**. For some reason, Mike always have a microphone in his hand. Weird.
  + Enable **alice** to use **sudo** command. She is a trusted user and will be allowed to do anything she wants with sudo.
  + Enable **mike** to use **sudo** as well. He’s kind of suspect, but he is from accounting, so we will let him use the **find** command and maybe a few others.
* Along the way, we’ll learn some useful stuff like:
  + How to create a new user from the shell.
  + How to change a user’s password from the shell.

# How Do You Do, Sudo?

Historically, people used the **su** command to turn ourselves into root. In practice, running **su** to become root is risky because it requires the user to know the root password. If you have multiple people who need to have root-level access to a server, this means giving the same password to each of them. Do you want them all to know the root password? If you do that, imagine how many servers’ passwords you might have to change passwords every time an administrator leaves your company. Not good.

Also, if all the administrators know the root password, they can log in as root directly (without using **su** at all) and then we will have no record in the logs of who was actually acting as root. Also not good.

Finally, the *principle of least privilege* is important. This principle means that we should have superuser permissions only when needed, and be logged in as a normal user otherwise.

So is it possible to authorize as root in a logged (accountable) way, without knowing the root password?

I’m glad you asked. The **sudo** command allows you to run one command as another user. If that command is a shell (such as **bash**), then the whole shell session will be run as that other user – just like using **su**. To set up **sudo**, you have to edit the file **/etc/sudoers**. For now, we only want to add one configuration entry, that will let your account run any command as any other user. Here’s what to do:

* Open a Terminal window.
* Become root and open the sudoers file by running these commands:
  + **su - root**
    - Supply the root password when prompted.
  + **export SUDO\_EDITOR=/usr/bin/gedit**
  + **sudoedit /etc/sudoers**
    - The **sudoedit** command allows you to edit the file you specify by using whatever editor is specified in the **SUDO\_EDITOR** system variable. We are using gedit since it’s very easy to use.

Since you are root when you launch sudoedit, sudoedit (meaning gedit) will also run as root. This will allow you to edit the sudoers file. Otherwise, you would not be able to even read it, since it has a mode of 440 (r--r-----).

* Once gedit launches, do the following:
  + Go to the end of the file.
  + Add a new line this one, except replace <username> with the name of your Linux user:
    - <username> ALL=(ALL) ALL

For example, on my computer, I entered “chris ALL=(ALL) ALL”. This means that the user chris will be allowed to run any command as any user while connected from any computer. The word ALL is a special keyword we can use instead of trying to actually list every possible computer, user, and command.

* + Save the file and exit gedit.
* Finally, to test that this works, do the following:
  + Return to your own session (stop being root) by running **exit** in your Terminal session. Your prompt should list your own username (instead of root) again.
  + As your own account, run this command, supplying YOUR password this time.
    - **sudo –u root /bin/bash**
  + This should give you access to a bash shell running as the user root. Usually I will use one of the two following shorter commands, which do the same thing:
    - **sudo su -i**
    - **sudo su - root**

# Adding a New User

Now let’s start setting things up for SSH. First we need to create a “guest” user on your computer. This is the account your partner will use when they connect via SSH. To do this, become root and run the following command, replacing <username> with your own username. For example, on my computer, I used chris\_guest instead of <username>\_guest.

* **useradd <username>\_guest**
* **passwd <username>\_guest**
  + Set the user’s password to **farmergiles** when prompted.

That’s right, *Princess Bride* (and *Spinal Tap*)fans – I am now Christopher Guest. If you don’t understand this comment, you can just ignore it.  
 For the best story ever that prominently features a blunderbuss, I recommend *Farmer Giles of Ham* by J. R. R. Tolkien. Nothing else comes close.

**What’s Next?**