Matt Favazza

Security Part 1

Ubuntu

1. To monitor connections to the server, we can check the **/var/log/auth.log** file. Here we see successful log ins, failed log ins, and IP addresses besides the ones from the server. Text

   Description automatically generated

We can further filter these out using grep. With the command **sudo grep "Failed password" /var/log/auth.log | grep "from"** we can specifically see failed attempts from an IP address.

Text

Description automatically generated

Here we can see a purposefully failed log in attempt I made, using an SSH client.

Now we can put this command in a script, add some other useful information and redirect the output of the commands into a text file.

A screenshot of a computer

Description automatically generated with medium confidence

And we can now run that to add every failed access attempt to a text file which we can monitor, or even send to ourselves via email. For now, I’m just going to leave it in a text file. We can also automate this processes too. Using cron, we can automatically run this script every day, hour, month etc and not have to worry about running it ourselves every time we log in. To do this we need to create a cron job.

To create a cron job we can type **cronjob -e**. Then select which editor you want to use. I’m using nano.

Text

Description automatically generated

And you will be presented with a screen like this.

Text

Description automatically generated

I found a useful comment to help me create the jobs easier, specifically for helping me with the time system.

Text

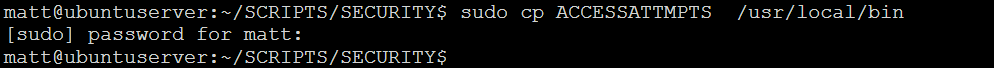
Description automatically generated

Now we can decide how often our script needs to be ran. I think daily is too often, so I will run it every week. To do this we need to edit the crontab file and add this line

Text

Description automatically generated

First off, we can see a different path from what I’ve been using. Usually I just have everything in the home directory, but now we see that the path is in the /user/local directory. I still want to keep a copy in my home directory, so we need to copy the script to the /usr/local/bin directory. To do this we can type **sudo cp ACCESSATTMPTS /usr/local/bin**



and if we switch to the new directory we can see the file was copied to there

Text

Description automatically generated

One last thing before we can call it good, is we need to now make it a .sh file. Again, Im going to copy the file into a new file called ACCESSATTMPTS.sh by typing **sudo cp ACCESTTATTPTS ACCESSATTMPTS.sh**

Graphical user interface, text

Description automatically generated

And now we can **sudo rm ACCESSATTMPTS** to leave this directory with the new file.

1. There are two directories I want to monitor. **/etc** and **/usr**. I’m choosing these two specifically because these are where most of the programs live on the server. Keeping an eye on installed programs can be useful, especially if multiple people use the system and are installing and removing programs. We can also see what programs have been documented to be installed or if something was installed and not documented, we can now take notice of that.

To do this is fairly simple. We can either get a hash sum or a **diff** result, but in this case, I want to have to only use a hash just to make things easier.

We can create a temporary tar file and check the sha1sum of it by using the command **echo tar cvf - /etc | sha1sum | sed '\/etc/d'.** This creates a file called ‘ – ‘ and runs the sha1sum to find the resulting hash. The sed command removes the output from the tar operation. We can put this into a script now for both directories.

Text

Description automatically generated

Here is the output

Text

Description automatically generated

Again I made this script in my SCRIPTS folder, so we need to move it to the /usr/local/bin.

**sudo cp DIRCHANGES.sh /usr/local/bin**

We can add this to the cronjob list. Type **crontab -e** and add thie script to the list

Text

Description automatically generated

I chose to run this once a week on Tuesdays.

1. We can monitor hidden files very easily. The **find** command will recursively search for anything with the given criteria. We can type **sudo find [dir] -name ".\*" -ls** and easily see everything we would want to know.

A computer screen capture

Description automatically generated with medium confidence

We can see the size of the file, the permissions, the author and last person to access it and the date.

I want to monitor the whole system for these hidden files, so the directories we need to monitor are : boot, dev, etc, home, lost+found, media, mnt, opt, proc, root, run, snap, srv, sys, usr, and var.

We can add all those to a script with the previous command with the output directed at a text file

Graphical user interface, text

Description automatically generated

This will generate an output of every hidden file on the system, which we can then manually filter out if we suspect there are hidden files on the system taking up unnecessary space or created by a user. I’m choosing the keep the roots in the output mostly because I would like to see everything that is hidden.

I don’t think this needs to be ran automatically, only when we suspect users of creating hidden files or if a storage gets too low, we can look at the hidden files to see if anything is taking up space.

Resources That helped

<https://opensource.com/article/17/11/how-use-cron-linux>

<https://www.networkworld.com/article/3598048/monitoring-failed-login-attempts-on-linux.html>

<https://tldp.org/LDP/sag/html/sag.html#ROOT-FS>

<https://stackoverflow.com/questions/545387/linux-compute-a-single-hash-for-a-given-folder-contents>

<https://www.cyberciti.biz/faq/unix-linux-centos-ubuntu-find-hidden-files-recursively/>