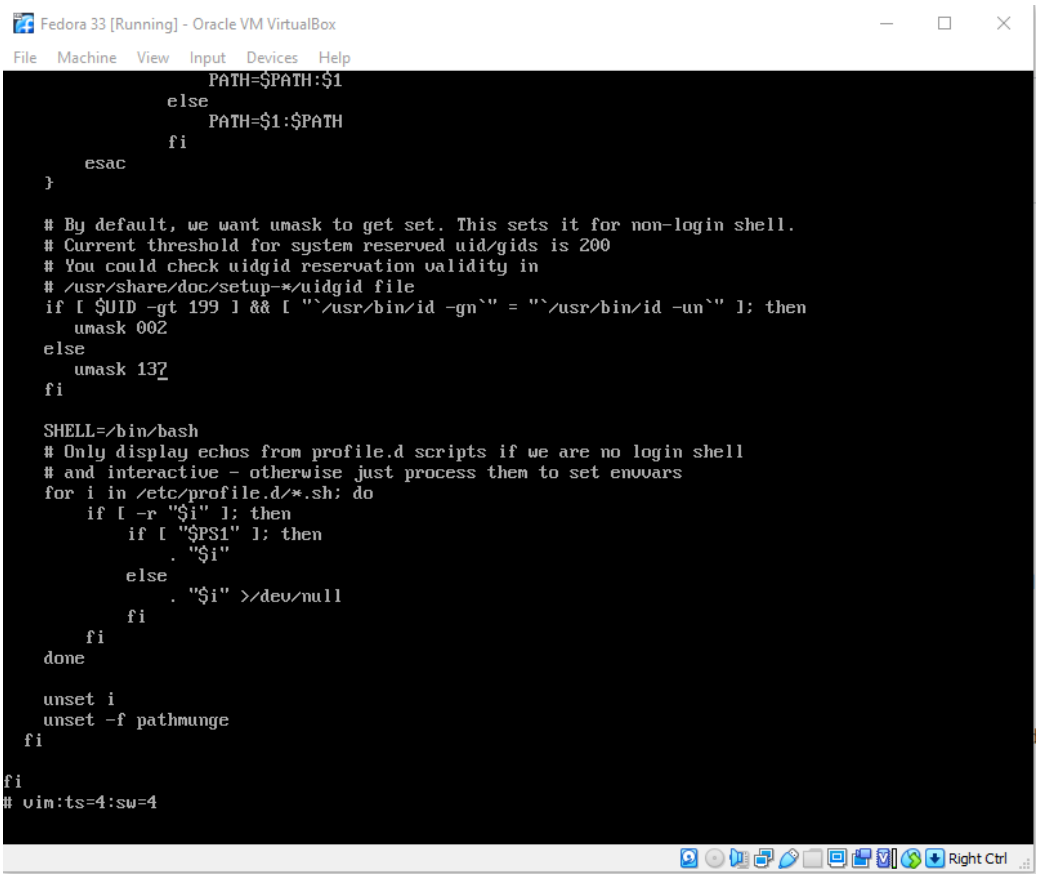
Issue #1: Default file creation permissions are not adequate.

For most of you, the current default permissions for newly created files is: rw-r--r--, Read/Write for the owner, Read for the Group, Read for Others. The Paranoids want the system default for all users to be Read/Write for the Owner, Read for the Group, and no permissions for Others, which would mean all new files get created with these permissions: rw-r-----. They want to make sure new files get created with these permissions every time any user logs in and without the user having to issue any commands.

1. First I did a little bit of research online and found out some information on the Umask command that seemed like it would be helpful.
2. First I logged into my root user account. I had attempted to use the normal file permissions in "umask 640" in order to set the default file permissions, however, this did not work.
3. I then found a helpful resource online that listed the needed values in order to achieve the outcome that I wanted. I needed to change the values to "umask 137". This grants the owner the ability to write and read, the group to read, and the no permission for others.
4. I logged on to my normal account and created a file to see if this command changes the permission for other users as well. Unfortunately, it did not.
5. I implemented the umask command on my normal account and it worked as well.
6. I decided to see if this was a permanent fix, so I logged out and logged back in on both my root and regular user account, but it was back to the default permission values.
7. After a while of looking online some more, I found a website that suggested changing the umask in the configuration file /etc/bashrc.
8. I did successfully change both umask lines in the file to the value of 137, but there was no difference in the default umask/permissions, as they were still the default of 002 and 022.
9. I tried to restart my Fedora VM to see if maybe I needed to restart the system after updating the configuration file, but that did not change anything either.
10. After more research, I found multiple other websites that stated changing only the second "umask" line should fix the issue, however, I still did not get the correct results. (Sources: https://www.cyberciti.biz/tips/understanding-linux-unix-umask-value-usage.html, and https://www.computernetworkingnotes.com/rhce-study-guide/how-to-change-default-umask-permission-in-linux.html).
11. I created a new user to see if the change only applies to new users, but however, it still does not work.

Changing the umask in /etc/bashrc:



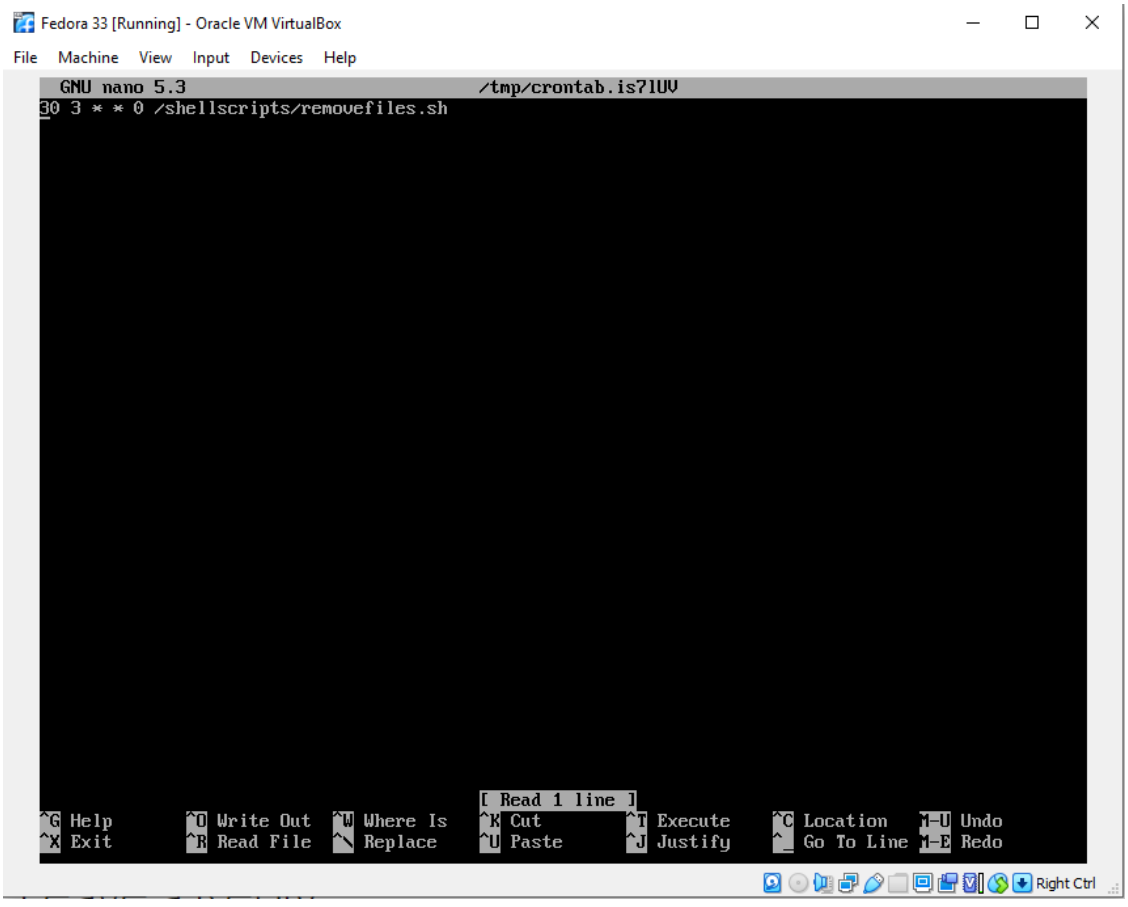
Finding #2: Older files are accumulating in the /tmp directory

All users can and do make use of the /tmp directory to create scratch files, work files and other temporary files. However, the Paranoids found some older files in /tmp with sensitive information. Set your system to automatically, once a week, on Sunday, at 3:30am, delete any files in the /tmp directory that are older than a week.

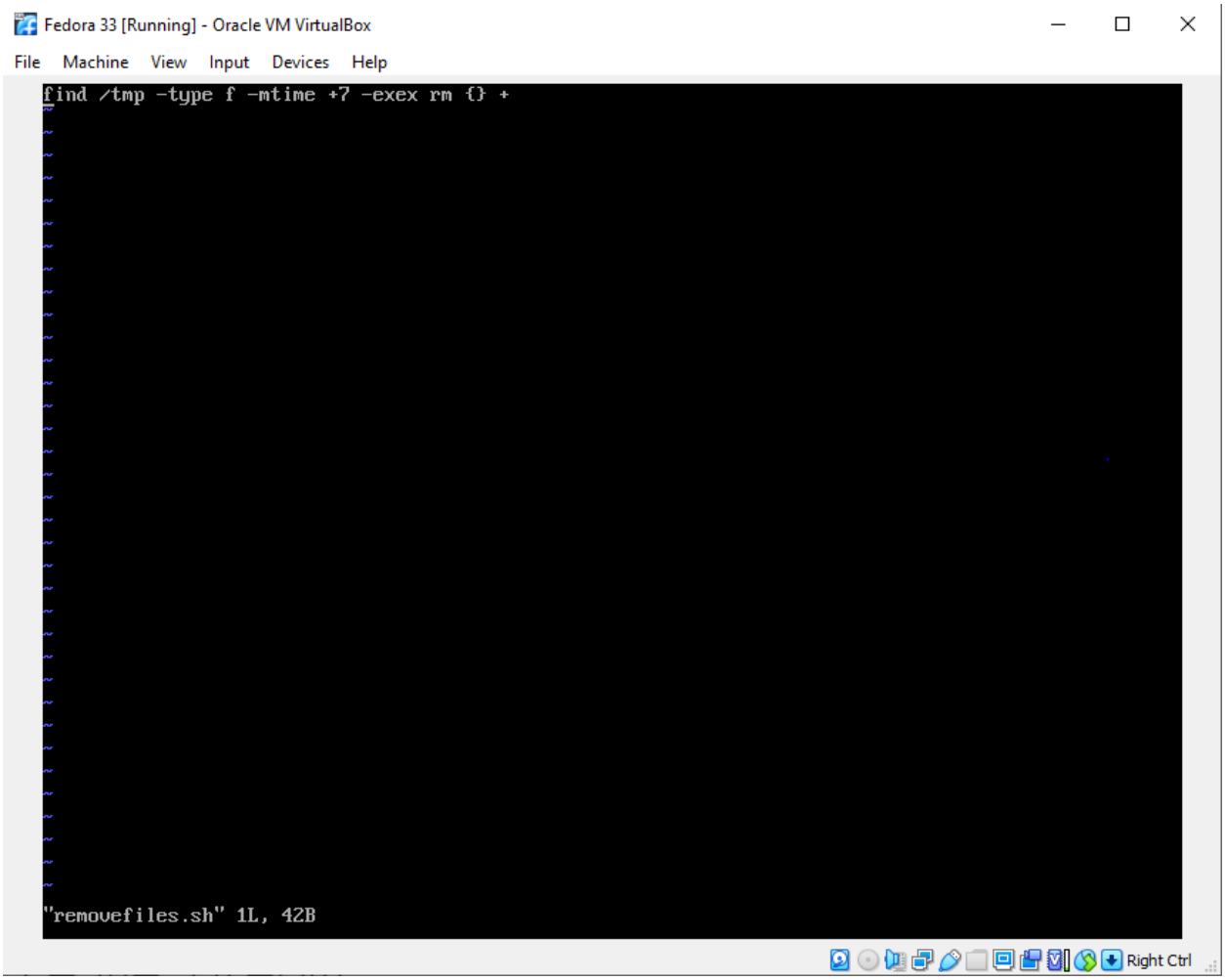
1. The first idea that I had was to go straight to the cron tables and attempt to create a schedule that would perform the expected outcome. I could figure out everything except for deleting files that are older than one week.
2. Once again, I referred to the Internet in order to gather more information.
3. I eventually found the suggestion to create a shell script to run that deletes the files every week. (Source: https://www.developerfiles.com/delete-files-on-linux-using-a-scheduled-cron-job/)
4. After taking this into consideration, I created a shellscript in my newly created “shellscripts” directory. I created the shellscript with the “touch removefile.sh command” and then “vi removefiles.sh”.

5. The given shell scrip is “find /test/my_folder -type f -mtime +7 -exec rm {} +”. I typed this into the shell script that I had just created, modifying the directory that the files need to be deleted in though. (Changing /test/my_folder to /tmp).
6. Next, I saved the shellscript and typed “crontab -e” in order to bring up my cron table.
7. I typed the command “30 3 * * 0 /shellscripts/removefiles.sh” into the file using the nano editor and then saved the file. The 30 represents how many minutes past the hour the scheduled command will execute. The 3 stands for 3:00 AM. The two asterisks mean every day of the month and every month of the year. Lastly the 0 stands for running the cron table every Sunday. Now, everything in the /tmp directory that is older than 7 days will get automatically deleted once a week, every Sunday, at 3:30 AM. Then the last segment of the cron table is what activates the shell script to delete the files.

The cron table:



The shell script:



Finding #3: Remote Access has to be manually enabled after each boot. Also, a suspected hacker IP address needs to be blocked.

The Paranoids note that you have to manually enable remote (ssh) logins after every reboot. This creates a potential issue if you, as root, type an incorrect command. Set up your system so that remote login via ssh is available after every reboot, without needing to issue any commands by hand. Also, they have noted that there are many, many failed logins from the host with the IP address of 172.0.15.12. Block this IP address from being able to login via SSH.

Problem #3 - Extra Credit

1. I decided to tackle the second part of this question first. I wasn't quite sure where to start, so I did a little research online, once again.
2. I came across a very informative article on how to block specific IP address from being able to log in via SSH. (Source: https://www.techrepublic.com/article/how-to-block-ssh-access-for-specific-ip-addresses/)

3. The first step is to create a file in the /etc directory called hosts.deny. This can be done with the command "vi /etc/hosts.deny".
4. Next, I typed "sshd 172.0.15.12" and saved the hosts.deny file.
5. In order for this file to take effect, the SSH daemon must be restarted with the command "systemctl restart sshd".
6. Now, whenever that IP attempts to login via SSH, they will not be able to, as their IP is blocked.

Finding #4: The system clock drifts making timestamps in security log files inaccurate.

After looking the system log files, they note that the time of some events in the log files doesn't match the time at which they performed the tests that triggered the event(s). They conclude that your system clock is not set up to sync with an external time reference service.

Do the needed steps to ensure the system clock on your Linux systems is always in sync with the actual time and any drifting is self-correcting.

1. The first thing that I did was do some research on the internet to see if there was any simple solution.
2. I did find a website that told me to run the command ntpdate 0.pool.ntp.org in order to update my system clock.
3. I ran the command and I needed to download and install the ntpdate packages, so I waited a little bit while it was performing that task.
4. After that, I ran the task again and it updated my system clock. I don't think this had any major influence on the outcome.
5. After running that command, I found a great resource on the Internet that provided a step by step tutorial on how to make sure your time is always is always in sync and self corrects based on NTP. (Source: https://tekneed.com/how-to-synchronize-configure-time-in-linux/)
6. The first step is to set the system hardware clock with the command "hwclock –systohc".
7. The next step is verify that the time zone information is correct with "timedatectl".
8. In the next step, I had to install the package using the command "yum install chrony".
9. Next, I verified that Chrony is running by using systemctl status chronyd.
10. Once again, I had to verify the time zone with "timedatectl status".
11. Next, I had to activate the NTP service with the command "timedatectl set-ntp yes".
12. The next step was to make sure that the NTP service was actually running by running the command "systemct1 status chronyd.service".
13. Next, I had to verify the time once again with the "timedatectl status" command.
14. The next step was to reset my time zone back to New York with "timedatectl set-timezone America/New_York".
15. Next, I can verify the correct time by running the "timedatectl status" command.
16. Now, my system should always be in sync and automatically correct the system clock by using the Network Time Protocol. You can see that automatic time synchronization is on under "system clock synchronized" if you run the command "timedatectl status". (Screenshot below)

Screenshot of synchronized time: