After running lynis on both systems, a quick scan of the results tells me that I may have a bit of work to do. The first thing that needs to be done for each system is a reboot. I also need to set expiration dates for passwords and it would probably be a good idea to install a professional malware scanner. The CentOS machine should also have updates automated. On the Ubuntu machine, I'm not sure how necessary it is, but it was suggested that I set a password for GRUB bootloader.

The script used to monitor system health follows

#creates a directory where this info will be stored

#if directory already exists, this will be skipped

mkdir -p health

#check subsystems

iostat > health/subsystems.log

#check memory

#I suppose you could also check this with "less /proc/meminfo"

cat /proc/meminfo > health/memstat.log

#check cpu

mpstat > health/cpu.log

#check processes

pstree > health/processes.log

#check network status

uptime > health/netstatus.log

#informs users where information can be found

echo 'Subsystems info can be found with "less health/subsystems.log".'

echo 'Memory info can be found with "less health/memstat.log".'

echo 'CPU info can be found with "less health/cpu.log".'

echo 'Processes info can be found with "less health/processes.log".'

echo 'Network status info can be found with "less health/netstatus.log".'

echo 'When you are done viewing info, press "q".'

Checking subsystems could be done using nmon instead of iostat. However, there is no guarantee that nmon is already installed. Another command in the script mpstat, and both that and iostat come together when sysstat is installed. With that in mind, I decided using iostat was the way to go.

Running the script locally is easy, just use the command ". monitor.sh". You will need to have permission to run the script. To give permission to run this script, use "chmod +x monitor.sh". This will give all users the permissions required to run the script. Using script remotely requires a bit of setup. Before you can run the script remotely, you will need to use a few commands. First, use "ssh-keygen -t rsa". You should then press enter and then you will be prompted for a passphrase, you do not need to enter one to proceed, but it might be a good idea. After pressing enter, enter the passphrase a second time. This will set up your public and private keys. After that, use "scp $HOME/.ssh/id\_rsa.pub [user]@[server]:~/.ssh/authorized\_keys" to copy the public key to the remote server. If everything has been done correctly, you then you can run the script remotely using "ssh [user]@[server] 'bash -s' < monitor.sh".

Sources

www.linuxsecrets.com/discussions/9998-installing-mpstat-stats

www.thegeekdiary.com/understanding-proc-meminfo-file-analyzing-memory-utilization-in-linux/

linux.die.net/man/1/pstree

www.geeksforgeeks.org/linux-uptime-command-with-examples/

askubuntu.com/questions/204065/how-to-run-local-shell-script-on-remote-server-via-ssh

www.cyberciti.biz/faq/how-to-set-up-ssh-keys-on-linux-unix/