For detecting ip addresses, I thought that tcpdump suited my needs fairly well. I wanted to set up a cron job that would execute the script continuously between midnight and six in the morning. I would consider attempts to access the server at these times an unusual activity. Access to the /etc/passwd file at these times would be especially concerning. However, I could not find a method where this would be possible. So I decided to set the script to use tcpdump and output the results to a file. Then it would wait for six hours before killing the process.

#should be set to run at midnight every night

#adding the date the script was run to the output file name allows keeping track of server access over multiple days

NOW=$(date +"%b-%d-%y")

tcpdump -q -w accesslog-$NOW.pcap &

pid=$!

#originally, I wanted to use cron to have a script job that runs continuously for six hours, but I could not get that to work as well

#there is probably a way to do that, but I am not sure what it is, so I'll settle for killing the process after six hours

sleep 6h

kill $pid

For checking differences in directories, I realized making a copy of the directory would be necessary. This would allow me to check differences between the original directory and its duplicate. But if the directory was changed at all, every single time the script was run, it would report the differences. To get around this, I would check for differences and then overwrite the duplicate directory with a new copy. The differences found would be output to a file that included the date in its name. By scheduling a job to run this script everyday, I would have a kind of report on the differences between the directory today and the day before.

#should be set to run daily, 6 AM is a good time for that

#including the date the script was run in the output file name makes it possible to track when changes were made from one day to another

NOW=$(date +"%b-%d-%y")

diff /var/log backup/log > changes-$NOW.txt

#before running the script the first time, a copy of /var/log should be created

#copying the directory allows the script to be run every day with new data to compare to

cp -r /var/log backup/log

Checking changes to hidden files gave me the most trouble. Eventually, I found I could use auditctl to monitor hidden files. However, this would not carry over on reboot. I then found I could make changes to the audit.rules file to make these rules apply on reboot. Here are the rules that would need to be added.

-w /. -p w -k main-check

-w /.. -p w -k second-check

-w /.bash\_logout -p w -k logout-check

-w /.bash\_profile -p w -k profile-check

-w /.bashrc -p w -k bashrc-check

-w /.cshrc -p w -k cshrc-check

-w /.ssh -p w -k ssh-check

-w /.tcshrc -p w -k tcshrc-check

I could then use ausearch to call these rules and use them to search through logs using the rules I had put in place. This would give me the time for each event as well as the user that issued the command. But this tended to take a while, so I decided to only check the logs for the previous day. Doing this meant that I would need to schedule a cron job to run the script every day, but it would reduced how long it would take for the searches to complete. I would then output the results of the search to a file with the date in its name.

#auditd can be used to record changes to hidden files and directories

#script should be set to run daily using cron, best time is at midnight

#We need to get the date first so we can add that to the file name at the end

NOW=$(date +"%b-%d-%y")

#Now we do a search based on the rules we set up

ausearch --input-logs -k main-check -ts yesterday -i -l > audit-$NOW.txt

ausearch --input-logs -k second-check -ts yesterday -i -l >> audit-$NOW.txt

ausearch --input-logs -k logout-check -ts yesterday -i -l >> audit-$NOW.txt

ausearch --input-logs -k profile-check -ts yesterday -i -l >> audit-$NOW.txt

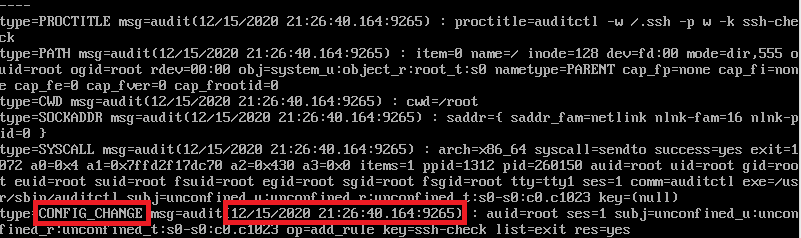
ausearch --input-logs -k bashrc-check -ts yesterday -i -l >> audit-$NOW.txt

ausearch --input-logs -k cshrc-check -ts yesterday -i -l >> audit-$NOW.txt

ausearch --input-logs -k ssh-check -ts yesterday -i -l >> audit-$NOW.txt

ausearch --input-logs -k tcshrc-check -ts yesterday -i -l >> audit-$NOW.txt

Here is a sample output of this script. It shows that a change of configuration occurred on 12/15/2020 at 9:26.



One thing I would like to have a cron job for is one to check for the existence of a script with permissions set to 777. This would make it a script that could be viewed, altered, and executed by anyone. Another I'd like to have a job for is checking for alterations to permission of files by anyone but root or the file owner. There is no reason for anyone who is not root or the file owner to be changing permissions to a file.

Sources

stackoverflow.com/questions/25731643/how-to-schedule-tcpdump-to-run-for-a-specific-period-of-time

www.geeksforgeeks.org/diff-command-linux-examples/

www.tecmint.com/query-audit-logs-using-ausearch-tool-on-centos-rhel/

Github link

https://github.com/RobertgSchultz8/scripts\_for\_security\_1