Lindsay Graham

Security Part 1

**Ubuntu Server**

**Example 1. Write a script to detect ip addresses trying to gain access, examples of things to pay attention to include all use between midnight and 6, all logins for a specific user, anything else you consider behavior that should send up a red flag. Make sure to include in your assignment what you consider a ”red flag” and why.**

To detect suspicious logins, I chose to write a Python script that would use the **lastlog** command to read from the /var/log/lastlog file. The **lastlog** command shows the latest logins for every system user (Stocker, 2015). When I looked at the output, I noticed that my last login said September 9th. I know that can’t be true because I log on almost daily. Then I recognized that the IP address in the ‘From’ column was that of my Kali Linux machine. I don’t understand why my daily logins aren’t there, and why this Kali one is.

**Figure 1.1** Results of **lastlog** command

A screenshot of a computer

Description automatically generated

I wrote a script in Python that used the lastlog command to send output to a variable. Then I parsed the lines to look for output from a specific user. If that user was found, there was a check to look for logins that were greater than or equal to 20 (8pm) and less than or equal to 6 (6am).

**Figure 1.2** Script to detect unusual login activity

A screenshot of a computer

Description automatically generated

I included a line that was the opposite of my criteria to test that it works. I commented out my original if statement that looked for logins greater than or equal to 20 or less than or equal to 6 with an if statement that looked for hours less than or equal to 20. The testing showed the scripts worked, but I also know that **lastlog** only shows one login for lag, and from a date in September, so I don’t trust that this script would work for the purposes of finding recent suspicious logins. As a result, I decided to pivot and copy my script, modifying the **lastlog** command to use the **last** command instead.

**Figure 1.3** Image showing the script testing

A screenshot of a computer

Description automatically generated

**Figure 1.4** Image showing the script testing



The **last** command shows recent logins that are stored in the /var/log/wtmp file, with the newest logins first.

**Figure 1.5** Image showing the results of the last command

A screenshot of a computer

Description automatically generated

**Figure 1.6** Image of Python script using the last command

A screenshot of a computer program

Description automatically generated

This script works, but also creates an error. The last command has a lot of output, and the lines are not all formatted the same way. I believe that I need to add something to my script that says to ignore all lines that do not start with “lag.” This needs to be investigated.

**Figure 1.7** Image showing the script testing with error

A screenshot of a computer

Description automatically generated

**Figure 1.8** Image showing the script testing with error

A screen shot of a computer program

Description automatically generated

I attempted to incorporate regular expressions in the script so that instead of searching for a username, it could be modified to search for an IP address. I wasn’t able to get it to work. Depending on the login entry, each row can be formatted differently and that was a bit of a challenge. A potential enhancement to the script could be to collect user input to search for a specific user’s activity. For this script, I assumed a traditional work environment where employees are usually logged off by 6 pm and don’t sign on until 7 or 8 am the next day. This script could help detect anomalous login activity which may be worth investigating. If a person’s normal work day ends at 5, and they are logging in at 2 am, it may signify that their credentials have been stolen and the login is not theirs. It could also be perfectly legitimate. Perhaps someone couldn’t sleep because they were stressed about work and figured they were up, so might as well get something done. If it was not detected, it could not be investigated, and that is why a script to detect anomalous behavior is important.

**Example 2. Write a script to detect changes to a specific directory**

I chose to use diff as a way to detect changes to the /var/log directory. First, I copied the log directory as log 2 (ServerFault, 2009).

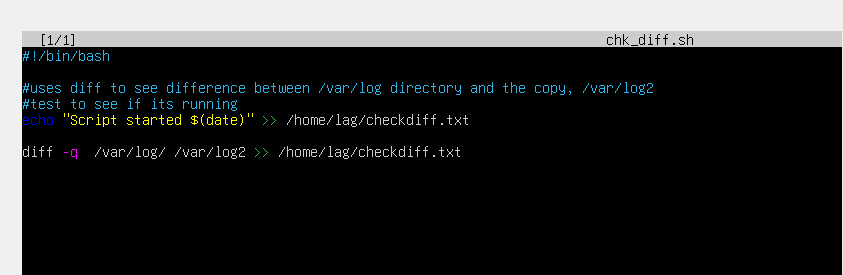
**Figure 2.1** Image showing the /var/log directory and the command to make a copy of the /var/log directory

A screenshot of a computer

Description automatically generated

Next, I wrote a basic script that uses diff to track any changes. Diff can analyze two files, or directories, and then print information that is different so that any new changes can be tracked Computer Hope (2024). For this example, diff compares the static copy, /var/log2 with the active /var/log directory and then sends the output to a file, checkdiff.txt. The -q option tells diff to only print output when files are different.

**Figure 2.2** Script that uses diff to compare two directories



In Linux, cron is used to schedule jobs or tasks at a defined cadence (Levinas, 2022). I created a new cron job to run the chk\_diff.sh script periodically. In the script shown in Figure 2.2, the first command is a simple test line to see if the cron job is running correctly. If it is, the line “Script started $(date)” will print to the text file checkdiff.txt. To use cron, I first had to install it with the command **sudo apt install cron**.

**Figure 2.3** Image of cron installation

A screen shot of a computer

Description automatically generated

After cron has installed, use the command crontab -e to open the crontab configuration file (Levinas, 2022). A prompt will appear that will let you select your preferred text editor. I selected option 1 for nano.

**Figure 2.4** Image of prompt that allows you to set text editor for crontab configuration file

A screenshot of a computer

Description automatically generated

After the comments, add a line for the cron job. The syntax for a cron job has six arguments:

[minute] [hour] [day of month] [month] [day of week] [command]

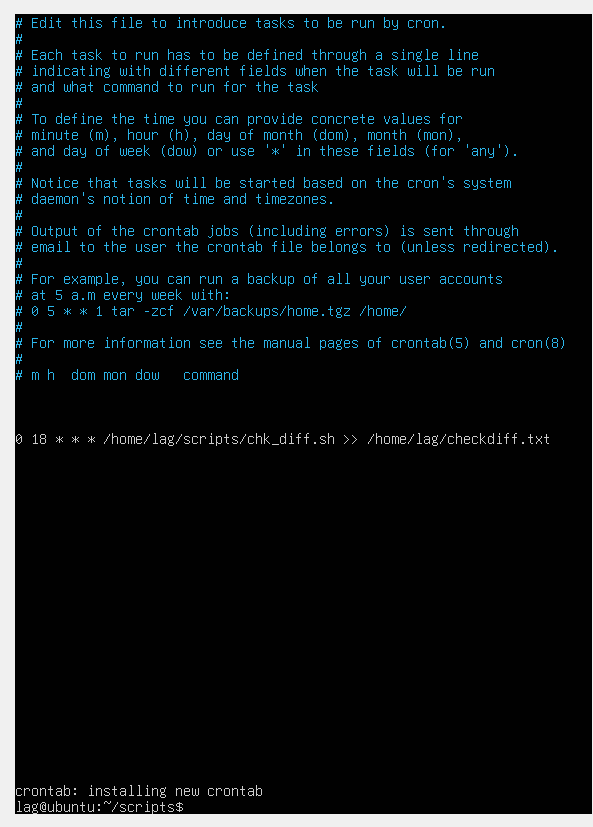
The minute, hour, and day of month arguments are numeric, with the time using a 24hr clock. The month and day of week can be specified as numbers or text. An asterisk can signify any value is accepted. There are other special values that can be specified such as setting a range, or step. When testing my cron job I used all astericks, which provided a lot of output. I then changed it to run only at 6pm (18).

The command for the cron job was:

0 18 \* \* \* /home/lag/scripts/chk\_diff.sh >> /home/lag/checkdiff.txt

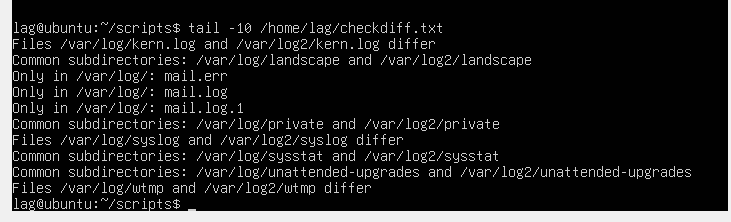
The /home/lag/scripts/chk\_diff.sh is the path that is needed to execute the chk\_diff.sh script. The /home/lag/checkdiff.txt specifies the file that should hold the output.

**Figure 2.5** Image of cron job configuration file



To see if the job was running, I used cat to view the contents of checkdiff.txt. There was a lot of information in the file because I had originally used all asterisks which resulted in it running every minute. Below, in Figure 2.6 the output shows the files that are not the same. It looks like most are log files, which would make sense because logs are constantly being generated.

**Figure 2.6** Last 10 lines of output



**Example 3.** **Monitor hidden files, root executables, and see if changes are made, who made them, and when they were changed.**

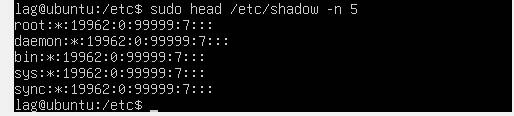
Two files that I chose to monitor were the hidden files, /etc/passwd and /etc/shadow. A hacker could potentially add a new user in this file, and if the file wasn’t regularly monitored, it could go undetected (Splunk Community, 2018). The /etc/passwd file has information about all user accounts such as user ID, group ID, and home directory (Zivanov, 2023). The information in the /etc/passwd file is in plain text, meaning that it can be read easily and is not encrypted. The /etc/shadow file is a related file that stores encrypted user passwords along with usernames and other information.

**Figure 3.1** Results showing entries in the /etc/passwd file

A screenshot of a computer program

Description automatically generated

**Figure 3.2** Results showing entries in the /etc/shadow file



The checksum (cksum) command in Linux can help verify a file’s integrity (Reynolds, 2013). The cksum shows results in three columns – the first shows the Cyclic Redundancy Check (CRC), the second shows the byte counts, and the third shows the files name that was checked. In the figure below, you can see the **cksum** results for the /etc/passwd and /etc/shadow files.

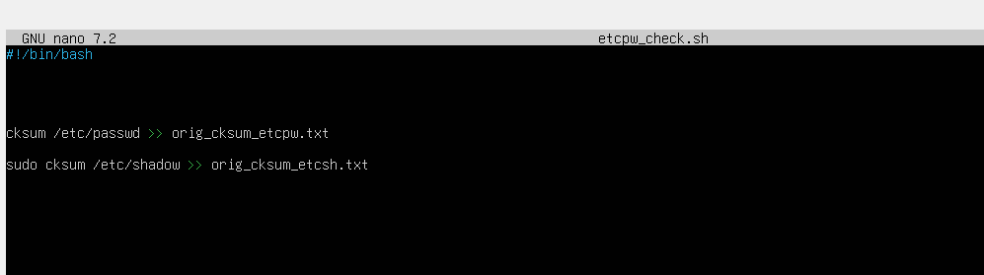
**Figure 3.3** Results of **cksum** command

A screenshot of a computer program

Description automatically generated

Next, I wrote a script that would run the checksums for those files and save them to a file where it would store the original checksums.

**Figure 3.4** Script that captures original checksums



Then, I created a script that used **diff** to rerun the checksums and compare with the original files.

**Figure 3.5** Script using diff to compare recent checksum with the original files

A screenshot of a computer program

Description automatically generated

Then, I set up a cron job to run the script periodically.

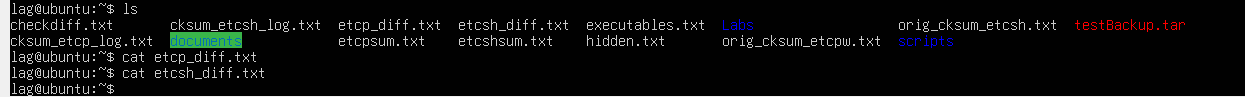
**Figure 3.6** New cron job to run the diff script

A screenshot of a computer

Description automatically generated

Next, I checked to verify that the etcp\_diff.txt and etcsh\_diff.txt files existed. Because nothing has been changed to the /etc/passwd and /etc/shadow files, when I used cat on the files there was no output.

**Figure 3.7** Results showing no output from the results of cat



To see if the checksums would show a change, I adecided to add a new user. I added a user hattie with the command **sudo useradd hattie** (Linuxize, 2018). Then, I added a password for hattie with the command sudo passwd hattie.

**Figure 3.8** Image showing the commands to add a new user

A screenshot of a computer program

Description automatically generated

I then checked the etcp\_diff.txt and etcsh\_diff.txt files again and saw that a change had been made. Below the results show that the files differed as well as the old and new checksums.

**Figure 3.9** Image showing the **diff** output files

A screenshot of a computer

Description automatically generated

**CentOS Server**

This section will show the same scripts and commands from the prior section, only run on the CentOS server.

**Example 1. Write a script to detect ip addresses trying to gain access, examples of things to pay attention to include all use between midnight and 6, all logins for a specific user, anything else you consider behavior that should send up a red flag. Make sure to include in your assignment what you consider a ”red flag” and why.**

To detect suspicious logins, I chose to write a Python script that would use the **lastlog** command to read from the /var/log/lastlog file. The **lastlog** command shows the latest logins for every system user (Stocker, 2015).

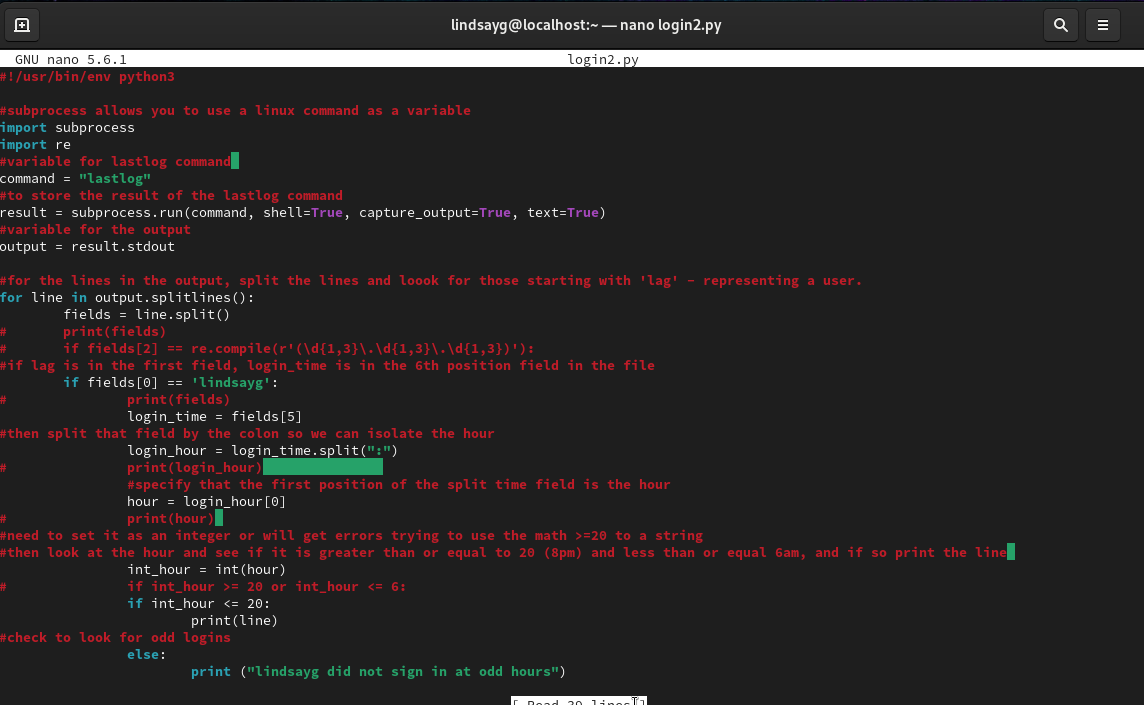
**Figure 1.1** Results of **lastlog** command

A screenshot of a computer

Description automatically generated

I wrote a script in Python that used the lastlog command to send output to a variable. Then I parsed the lines to look for output from a specific user. If that user was found, there was a check to look for logins that were greater than or equal to 20 (8pm) and less than or equal to 6 (6am). I only had to make a few small changes to get it to work. I needed to change the username it looked for and where the hour field was positioned in the line.

**Figure 1.2** Script to detect unusual login activity



I included a line that was the opposite of my criteria to test that it works. I commented out my original if statement that looked for logins greater than or equal to 20 or less than or equal to 6 with an if statement that looked for hours less than or equal to 20. The last login for lindsayg was at 10 am, so the results were the statement saying “lindsayg did not sign in at odd hours.” The second check was using the opposite criteria that looked for a login less than 20. This showed the 10 am login.

**Figure 1.3** Results of Testing

A black rectangular object with white text

Description automatically generated

The **last** command shows recent logins that are stored in the /var/log/wtmp file, with the newest logins first.

**Figure 1.5** Image showing the results of the last command

A screenshot of a computer

Description automatically generated

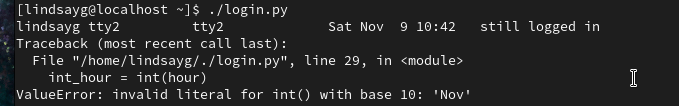
**Figure 1.6** Image of Python script using the last command

A screenshot of a computer program

Description automatically generated

This script works, but also creates an error. I believe that I need to further split the date time field out to remove the month.

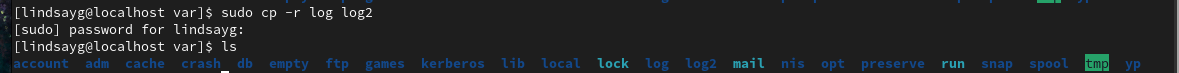
**Figure 1.7** Image showing the script testing with error



**Example 2. Write a script to detect changes to a specific directory**

I chose to use diff as a way to detect changes to the /var/log directory. First, I copied the log directory as log 2 (ServerFault, 2009).

**Figure 2.1** Image showing the /var/log directory and the command to make a copy of the /var/log directory



Next, I wrote a basic script that uses diff to track any changes. Diff can analyze two files, or directories, and then print information that is different so that any new changes can be tracked Computer Hope (2024). For this example, diff compares the static copy, /var/log2 with the active /var/log directory and then sends the output to a file, checkdiff.txt. The -q option tells diff to only print output when files are different.

**Figure 2.2** Script that uses diff to compare two directories

A screenshot of a computer

Description automatically generated

In CentOS 9, cronie is used to schedule jobs or tasks at a defined cadence (Uptimia, 2024). I wanted to create a new cron job to run the chk\_diff.sh script periodically. In the script shown in Figure 2.2, the first command is a simple test line to see if the cron job is running correctly. If it is, the line “Script started $(date)” will print to the text file checkdiff.txt. For CentOS 9, I checked to see if cronie was already installed and it was.

**Figure 2.3** Image showing cronie already installed

A computer screen with white text

Description automatically generated

After cron has installed, use the command crontab -e to open the crontab configuration file (Levinas, 2022). /The syntax for a cron job has six arguments: [minute] [hour] [day of month] [month] [day of week] [command]

The minute, hour, and day of month arguments are numeric, with the time using a 24hr clock. The month and day of week can be specified as numbers or text. An asterisk can signify any value is accepted. There are other special values that can be specified such as setting a range, or step. I used all astericks so that I could ensure it worked.

The command for the cron job was: \* \* \* \* \* /home/lindsayg/scripts/chk\_diff.sh >> /home/lindsayg/checkdiff.txt

The /home/lindsayg/scripts/chk\_diff.sh is the path that is needed to execute the chk\_diff.sh script. The /home/lindsayg/checkdiff.txt specifies the file that should hold the output.

**Figure 2.4** Image of cron job configuration file

A screenshot of a computer

Description automatically generated

To see if the job was running, I used tail to view a snippet of the contents of checkdiff.txt.

**Figure 2.5** Last 10 lines of output

A screen shot of a computer

Description automatically generated

**Example 3.** **Monitor hidden files, root executables, and see if changes are made, who made them, and when they were changed.**

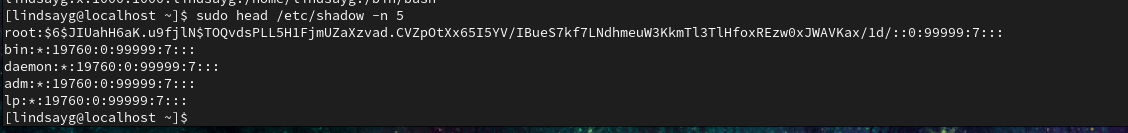
Two files that I chose to monitor were the hidden files, /etc/passwd and /etc/shadow. A hacker could potentially add a new user in this file, and if the file wasn’t regularly monitored, it could go undetected (Splunk Community, 2018). The /etc/passwd file has information about all user accounts such as user ID, group ID, and home directory (Zivanov, 2023). The information in the /etc/passwd file is in plain text, meaning that it can be read easily and is not encrypted. The /etc/shadow file is a related file that stores encrypted user passwords along with usernames and other information.

**Figure 3.1** Results showing entries in the /etc/passwd file

A screen shot of a computer

Description automatically generated

**Figure 3.2** Results showing entries in the /etc/shadow file



The checksum (cksum) command in Linux can help verify a file’s integrity (Reynolds, 2013). The cksum shows results in three columns – the first shows the Cyclic Redundancy Check (CRC), the second shows the byte counts, and the third shows the files name that was checked. In the figure below, you can see the **cksum** results for the /etc/passwd and /etc/shadow files.

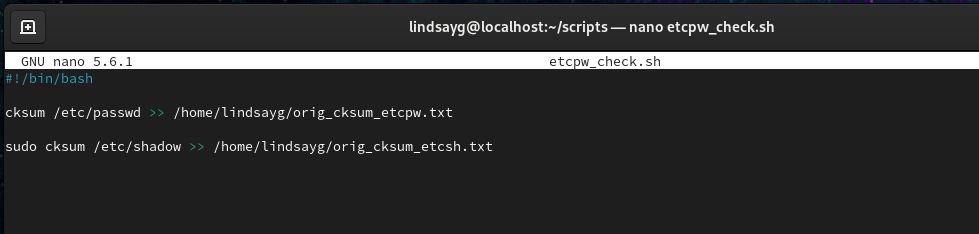
**Figure 3.3** Results of **cksum** command

A screen shot of a computer

Description automatically generated

Next, I wrote a script that would run the checksums for those files and save them to a file where it would store the original checksums.

**Figure 3.4** Script that captures original checksums



Then, I created a script that used **diff** to rerun the checksums and compare with the original files.

**Figure 3.5** Script using diff to compare recent checksum with the original files

A screenshot of a computer

Description automatically generated

Then, I set up a cron job to run the script periodically.

**Figure 3.6** New cron job to run the diff script

A screenshot of a computer

Description automatically generated

Next, I checked to verify that the etcp\_diff.txt and etcsh\_diff.txt files existed. Because nothing has been changed to the /etc/passwd and /etc/shadow files, when I used cat on the files there was no output other than the date checked.

**Figure 3.7** Results showing no output from the results of cat

A screenshot of a phone

Description automatically generated

To see if the checksums would show a change, I decided to add a new user. I added a user hattie with the command **sudo useradd hattie** (Linuxize, 2018). Then, I added a password for hattie with the command **sudo passwd hattie**.

**Figure 3.8** Image showing the commands to add a new user

A screenshot of a computer

Description automatically generated

My diff script is not working. I deleted the hattie user, and compared checksums in the terminal. I’m not sure what is wrong, but I will need to investigate.

**Figure 3.9** Image showing the different checksums

A screen shot of a computer screen

Description automatically generated

**REVISION**

**Example 3-NEW.** Monitor hidden files, root executables, and see if changes are made, who made them, and when they were changed

In my first attempt, I used a diff script that didn’t work for one of my servers and wasn’t really providing me with the information that I wanted. For this revised attempt, I used MD5 hashes and a python script to check for file integrity. First, I generated the hashes for the /etc/passwd and /etc/shadow files in the terminal using the command **md5sum ‘filename’** (GeeksforGeeks, 2024).

**Figure 3N.1** Image of MD5 sum hash being generated for /etc/passwd and /etc/shadow in the Ubuntu server

A computer screen with white text

Description automatically generated

**Figure 3N.2** Image of MD5 sum hash being generated for /etc/passwd and /etc/shadow in the CentOS server

A screenshot of a computer

Description automatically generated

Next, I found some helpful resources for how to write a Python script to generate the MD5 hash for /etc/passwd and /etc/shadow and compare it to the original hashes that I created in the terminal. At first, I attempted to use the command/subprocess for the MD5 command as I had with last in an earlier script. That didn’t work as I had hoped. Next, I used a sample script that I found in Stack Overflow as a guide for how to use **hashlib** which is similar to **cksum**. The script uses variables for the original hashes, and the files to check (StackOverflow, 2013). Then the file is read in binary and an MD5 sum is generated and stored as a variable. An IF/THEN clause checks to see if the original and generated hashes are the same and prints the findings to the terminal. I kept getting an error related to the line in the script that uses hashlib and hexdigest(). After much troubleshooting, it turned out that it was a typo in my code. I was writing hexidigest() instead of hexdigest(). Once I fixed that typo, the scripts ran as expected.

**Scripts**

**Figure 3N.3** Image of hash\_check.py script in the CentOS server

A screenshot of a computer program

Description automatically generated

A black background with green text

Description automatically generated

**Figure 3N.3** Image of hash\_check.py script in the Ubuntu server

A screenshot of a computer program

Description automatically generated

To test the scripts, I added a new user to both servers. I had already created an account for my older dog, so this time I made one for my younger dog. I added the user, reran the script, and confirmed that the hash had changed – indicating the changes in /etc/passwd and /etc/shadow.

**Figure 3N.4** Image showing the script being run before and after the user add in the CentOS server

A screenshot of a computer

Description automatically generated

**Figure 3N.5** Image showing the script being run before and after the user add in the Ubuntu server

A screenshot of a computer program

Description automatically generated

**References**

Computer Hope. (2024, September 10). *Linux diff command help and examples*. Www.computerhope.com. https://www.computerhope.com/unix/udiff.htm

GeeksforGeeks. (2024, July 19). *md5sum Command in Linux with Examples*. GeeksforGeeks. https://www.geeksforgeeks.org/md5sum-linux-command/

Levinas, M. (2022, January 26). *How to Use Cron to Automate Linux Jobs on Ubuntu 20.04 | Cherry Servers*. Cherry Servers. https://www.cherryservers.com/blog/how-to-use-cron-to-automate-linux-jobs-on-ubuntu-20-04#setup-your-first-cron-job

Linuxize. (2018, December 11). *How to Create Users in Linux (useradd Command)*. Linuxize.com; Linuxize. https://linuxize.com/post/how-to-create-users-in-linux-using-the-useradd-command/

Reynolds, L. (2013, March 21). *cksum command in Linux with examples*. LinuxConfig. https://linuxconfig.org/cksum

ServerFault. (2009, April 30). *Shell command to monitor changes in a file*. Server Fault. https://serverfault.com/questions/1669/shell-command-to-monitor-changes-in-a-file

Splunk Community. (2018, March 15). *Most critical files that must be monitoring on Linux in terms of security*. Community.splunk.com. https://community.splunk.com/t5/Deployment-Architecture/Most-critical-files-that-must-be-monitoring-on-Linux-in-terms-of/m-p/360579

StackOverflow. (2013, June 1). *How to calculate the MD5 checksum of a file in Python?* Stack Overflow. https://stackoverflow.com/questions/16874598/how-to-calculate-the-md5-checksum-of-a-file-in-python

Stocker, S. H. (2015, April 27). *Checking last logins with lastlog*. Computerworld. https://www.computerworld.com/article/1365243/checking-last-logins-with-lastlog.html

Stocker, S. H. (2018, March 20). *Reviewing logins on Linux*. Network World. https://www.networkworld.com/article/965376/reviewing-logins-on-linux.html

Uptimia. (2024). *How to Install Crontab on CentOS/RHEL Linux?* Uptimia.com; Uptimia. https://www.uptimia.com/questions/how-to-install-crontab-on-centos-rhel-linux

Zivanov, S. (2023, July 13). */etc/passwd File: What It Is & How to Use It*. Knowledge Base by PhoenixNAP. https://phoenixnap.com/kb/etc-passwd