Lindsay Graham

Linux Administration

<u>Linys System Audit</u>

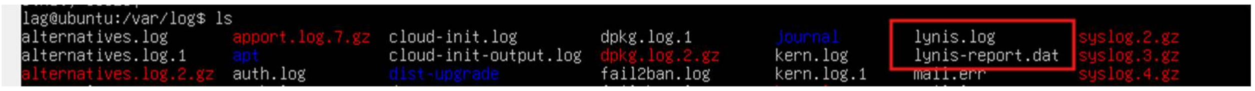
Linys performs a health scan to support system hardening (CISOfy, 2013). Hardening is a way to strengthen a system's defense. Other uses are security auditing, compliance testing, penetration testing, and vulnerability detection. Linys will provide a report in the terminal and will also create two files that will have the audit details (Hogan, 2017). One is a detailed log (lynis.log) file and the other is a general report (lynis-report.dat) file. Each scan will overwrite a prior report and log, so if you want to save it for historical purposes you must copy or save it before running the next audit. Please note that Lynis can be run in both privileged and non-privileged mode. I chose to run the scans in the privileged mode, because otherwise some scans may have been skipped.

A Lynis audit tests a system through various categories and then provides results, warnings, and suggestions. This report will summarize some of the findings for both the Ubuntu and CentOS servers.

Ubuntu Server

To install Linys on the Ubuntu Server use the command **sudo apt install lynis**. Then, to run Lynis use the command **sudo lynis audit system** (CISOfy, n.d.). After running Lynis, results will be shown in the terminal and in the log and report files that were created in the /var/log directory.

Figure 1.1 Image showing the Lynis log and report files



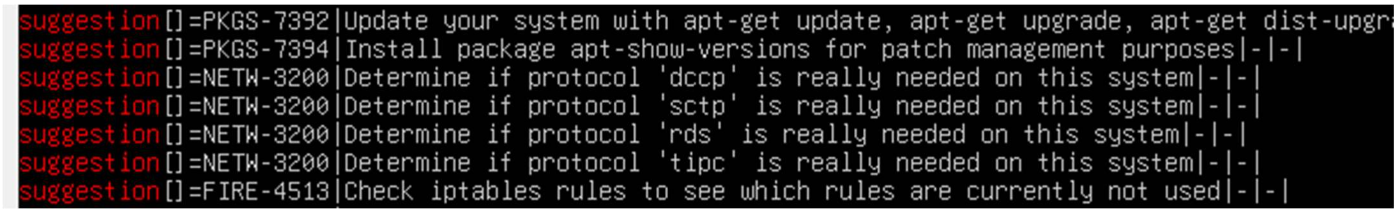
```
lag@ubuntu:/var/log$ ls
alternatives.log        apport.log.7.gz  cloud-init.log         dpkg.log.1       journal    lynis.log          syslog.2.gz
alternatives.log.1      apt              cloud-init-output.log  dpkg.log.2.gz    kern.log   lynis-report.dat   syslog.3.gz
alternatives.log.2.gz   auth.log         dist-upgrade           fail2ban.log     kern.log.1 mail.err           syslog.4.gz
```

First, I chose to focus on the suggestions that came out of the report. One of the suggestions was that old or removed packages needed to be purged. I know that at various points I uninstalled and removed packages. For example, I installed Wireshark on this server and then uninstalled it to use Tshark instead. These old/removed programs are taking up space on the system, and Lynis suggested how to clean things up using the **aptitude purge** or **dpkg –purge** commands. Another suggestion was to set a password on the GRUB boot loader to prevent altering the boot configuration. This means that by setting a password, the server is protected from unauthorized modifications or booting (Red Hat, 2024).

There were three suggestions in the report that questioned whether the dccp, sctp, rds, and tipc protocols were needed on the system. I researched those protocols and discovered that the Center for Internet Security (CIS) benchmarks for Linux systems advise that dccp, sctp, rds, and tipc are uncommon network protocols and should be disabled in the kernel (Center for Internet Security, 2016). Disabling unused protocols can help limit the server's attack surface, by lessening the number of ways that it could be accessed by a potential hacker.

Other suggestions included setting minimum and maximum ages and expiration dates for passwords. Currently, there are only two users on this server- myself and a user named Hattie (after one of my dogs). The National Institute for Standards and Technology (NIST) password guidelines recommend only changing passwords if something goes wrong, which is a change from the traditional method of requiring password changes every few months (Mayers, 2024). Because it can be challenging for humans to remember long, complex passwords, having to frequently create and remember new ones (if not using a password manager) can result in weaker passwords. At this time, I would likely not add a minimum or maximum age for the passwords on this server.

Figure 1.2 Image showing a snippet of the suggestions from the Lynis audit

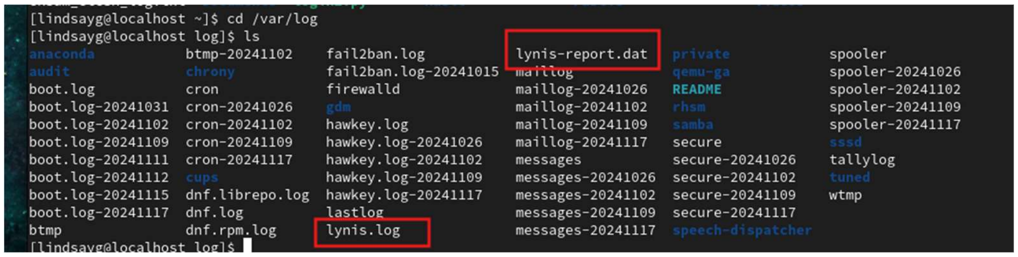


```
suggestion[]=PKGS-7392|Update your system with apt-get update, apt-get upgrade, apt-get dist-upgr
suggestion[]=PKGS-7394|Install package apt-show-versions for patch management purposes|-|-|
suggestion[]=NETW-3200|Determine if protocol 'dccp' is really needed on this system|-|-|
suggestion[]=NETW-3200|Determine if protocol 'sctp' is really needed on this system|-|-|
suggestion[]=NETW-3200|Determine if protocol 'rds' is really needed on this system|-|-|
suggestion[]=NETW-3200|Determine if protocol 'tipc' is really needed on this system|-|-|
suggestion[]=FIRE-4513|Check iptables rules to see which rules are currently not used|-|-|
```

CentOS Server

To install Linys on the CentOS Server use the command **sudo yum install lynis**. Then, to run Lynis use the command **sudo lynis audit system** (CISOfy, n.d.). After running Lynis, results will be shown in the terminal and in the log and report files that were created in the /var/log directory.

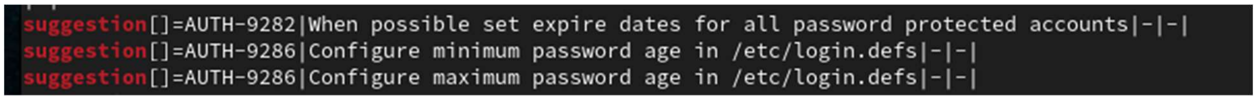
Figure 2.1 Image showing the Lynis log and report files



```
[lindsayg@localhost ~]$ cd /var/log
[lindsayg@localhost log]$ ls
anaconda          btmp-20241102    fail2ban.log          lynis-report.dat   private           spooler
audit             chrony           fail2ban.log-20241015 maillog            qemu-ga           spooler-20241026
boot.log          cron             firewalld             maillog-20241026   README            spooler-20241102
boot.log-20241031 cron-20241026    gdm                   maillog-20241102   rhsm              spooler-20241109
boot.log-20241102 cron-20241102    hawkey.log            maillog-20241109   samba             spooler-20241117
boot.log-20241109 cron-20241109    hawkey.log-20241026   maillog-20241117   secure            sssd
boot.log-20241111 cron-20241117    hawkey.log-20241102   messages           secure-20241026   tallylog
boot.log-20241112 cups             hawkey.log-20241109   messages-20241026  secure-20241102   tuned
boot.log-20241115 dnf.librepo.log  hawkey.log-20241117   messages-20241102  secure-20241109   wtmp
boot.log-20241117 dnf.log          lastlog               messages-20241109  secure-20241117
btmp              dnf.rpm.log      lynis.log             messages-20241117  speech-dispatcher
[lindsayg@localhost log]$
```

Like with the Ubuntu server, I chose to check the suggestions that were given as a result. Like the Ubuntu server, there were warnings about password age and expiration dates.

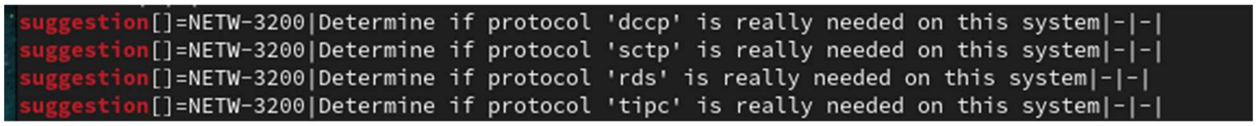
Figure 2.2 Image showing the suggestions about password age and expiration dates



```
suggestion[]=AUTH-9282|When possible set expire dates for all password protected accounts|-|-|
suggestion[]=AUTH-9286|Configure minimum password age in /etc/login.defs|-|-|
suggestion[]=AUTH-9286|Configure maximum password age in /etc/login.defs|-|-|
```

Like the Ubuntu server, it was also suggested to determine whether dccp, sctp, rds, and tipc were really needed.

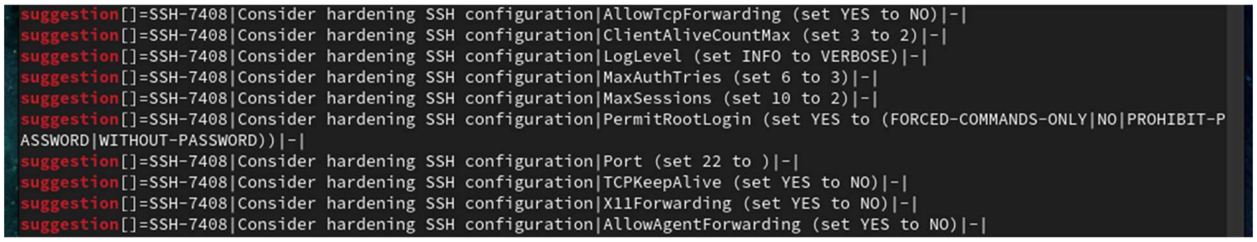
Figure 2.3 Image showing the suggestions about the uncommon network protocols



```
suggestion[]=NETW-3200|Determine if protocol 'dccp' is really needed on this system|-|-|
suggestion[]=NETW-3200|Determine if protocol 'sctp' is really needed on this system|-|-|
suggestion[]=NETW-3200|Determine if protocol 'rds' is really needed on this system|-|-|
suggestion[]=NETW-3200|Determine if protocol 'tipc' is really needed on this system|-|-|
```

There were quite a few suggestions related to the SSH configuration.

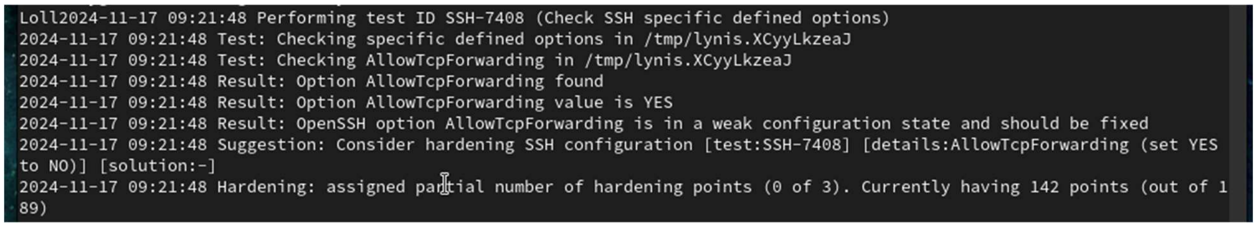
Figure 2.4 Image showing the SSH suggestions



```
suggestion[]=SSH-7408|Consider hardening SSH configuration|AllowTcpForwarding (set YES to NO)|-|
suggestion[]=SSH-7408|Consider hardening SSH configuration|ClientAliveCountMax (set 3 to 2)|-|
suggestion[]=SSH-7408|Consider hardening SSH configuration|LogLevel (set INFO to VERBOSE)|-|
suggestion[]=SSH-7408|Consider hardening SSH configuration|MaxAuthTries (set 6 to 3)|-|
suggestion[]=SSH-7408|Consider hardening SSH configuration|MaxSessions (set 10 to 2)|-|
suggestion[]=SSH-7408|Consider hardening SSH configuration|PermitRootLogin (set YES to (FORCED-COMMANDS-ONLY|NO|PROHIBIT-P
ASSWORD|WITHOUT-PASSWORD))|-|
suggestion[]=SSH-7408|Consider hardening SSH configuration|Port (set 22 to )|-|
suggestion[]=SSH-7408|Consider hardening SSH configuration|TCPKeepAlive (set YES to NO)|-|
suggestion[]=SSH-7408|Consider hardening SSH configuration|X11Forwarding (set YES to NO)|-|
suggestion[]=SSH-7408|Consider hardening SSH configuration|AllowAgentForwarding (set YES to NO)|-|
```

To see more information related to the specific test on SSH, I used the command **sudo lynis show details SSH-7408** (i.e., test-id). This provided additional information about the test that may help explain the suggestions better. In looking at the results in more detail, you can gain a better understanding of the suggestions. For example, the first test stated that the AllowTcpForwarding with OpenSSH was a weak configuration and should be fixed.

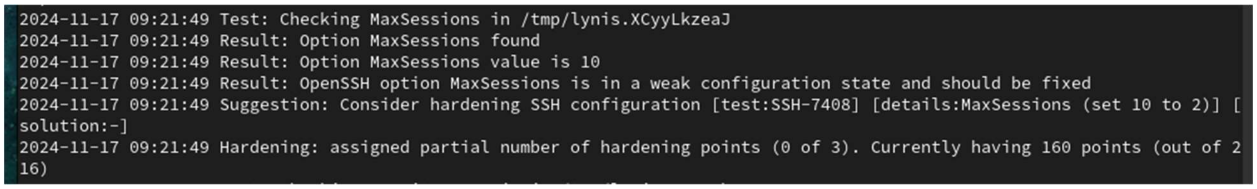
Figure 2.5 Snippet of results showing additional details on the OpenSSH configuration



```
Loll2024-11-17 09:21:48 Performing test ID SSH-7408 (Check SSH specific defined options)
2024-11-17 09:21:48 Test: Checking specific defined options in /tmp/lynis.XCyyLkzeaJ
2024-11-17 09:21:48 Test: Checking AllowTcpForwarding in /tmp/lynis.XCyyLkzeaJ
2024-11-17 09:21:48 Result: Option AllowTcpForwarding found
2024-11-17 09:21:48 Result: Option AllowTcpForwarding value is YES
2024-11-17 09:21:48 Result: OpenSSH option AllowTcpForwarding is in a weak configuration state and should be fixed
2024-11-17 09:21:48 Suggestion: Consider hardening SSH configuration [test:SSH-7408] [details:AllowTcpForwarding (set YES
to NO)] [solution:-]
2024-11-17 09:21:48 Hardening: assigned partial number of hardening points (0 of 3). Currently having 142 points (out of 1
89)
```

Another suggestion was that the OpenSSH MaxSessions value is 10 which is not advised. The suggestion was to change that value to 2 to be more secure.

Figure 2.6 Snippet of results showing the test on SSH MaxSessions and suggestion



```
2024-11-17 09:21:49 Test: Checking MaxSessions in /tmp/lynis.XCyyLkzeaJ
2024-11-17 09:21:49 Result: Option MaxSessions found
2024-11-17 09:21:49 Result: Option MaxSessions value is 10
2024-11-17 09:21:49 Result: OpenSSH option MaxSessions is in a weak configuration state and should be fixed
2024-11-17 09:21:49 Suggestion: Consider hardening SSH configuration [test:SSH-7408] [details:MaxSessions (set 10 to 2)] [
solution:-]
2024-11-17 09:21:49 Hardening: assigned partial number of hardening points (0 of 3). Currently having 160 points (out of 2
16)
```

Overall, the Lynis audit report is a helpful tool for assessing a system's security. The report has a high level of detail and offers concrete suggestions that are aligned with best practice. This can be very useful for hardening systems, limiting their attack surfaces, and staying in compliance with industry standards such as the CIS Benchmarks.

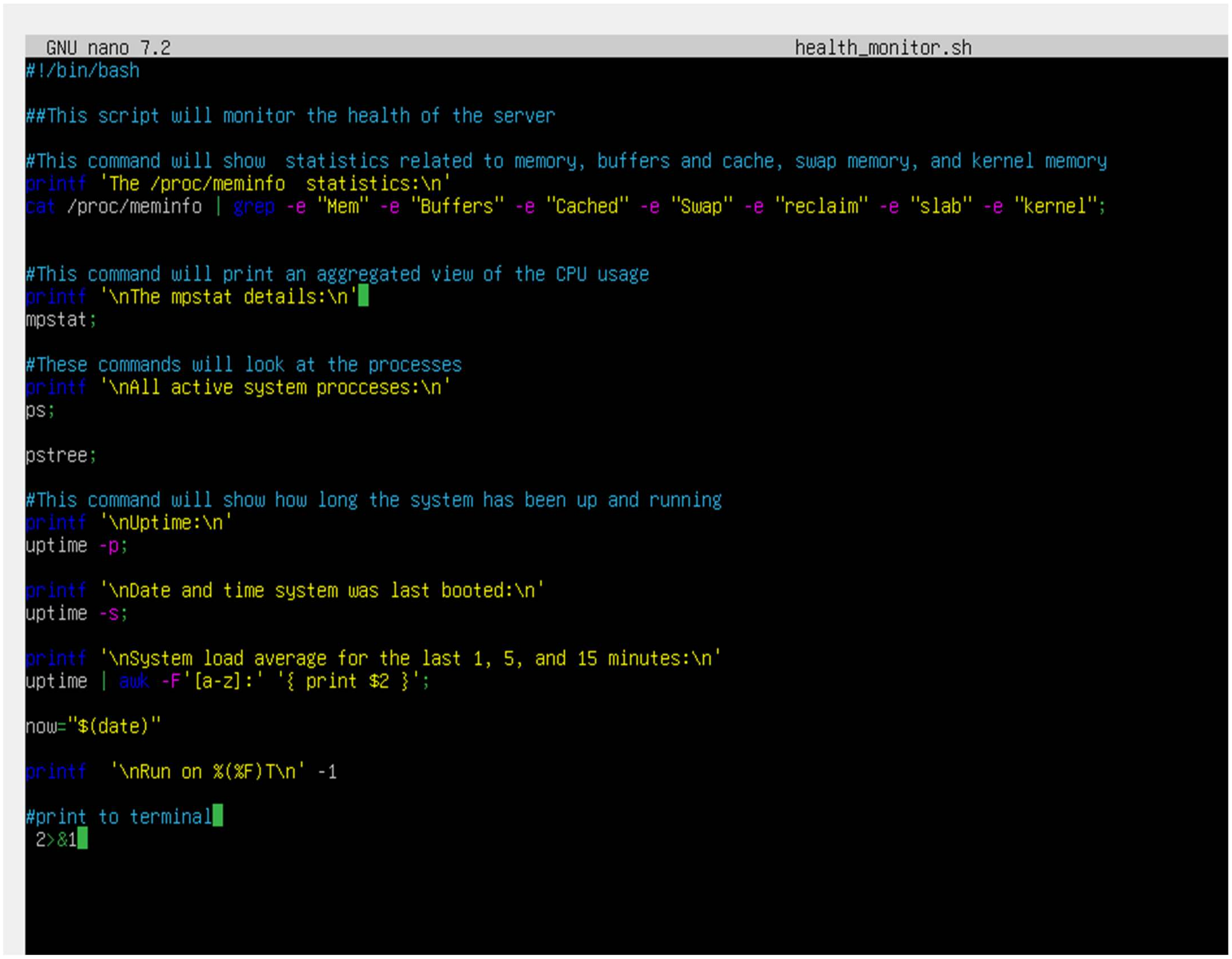
Health Monitoring

The /proc/meminfo file provides details about how memory is used on the system (Baeldung, 2024). Inside the files are details related to things like used and available memory, swap space, cache, and buffers. The following elements from the meminfo file were included in the script because of the insights into system memory that they provide. The total memory (MemTotal) shows the total usable RAM, the free memory (MemFree) shows the free RAM that is not being used, and the memory available (MemAvailable), shows the amount of RAM that can be allocated for any process. According to Baeldung (2024), “buffers are temporary storage components for process input and output” and cache “stores some data to serve future requests faster.” Buffers don’t increase processing size, but cache does so it could be helpful to keep a pulse on those statistics. Swap space is the memory that a system uses is the physical memory is full. Looking at the swap space is another way to keep tabs on how much memory is available on a system. This file also shows the amount of inactive and active memory which is helpful for understanding how systems handle memory. Kernel memory statistics are important because if kernel memory is depleted, processes could stop running.

The **mpstat** command shows the server's aggregate CPU usage statistics (GeeksforGeeks, 2020). The basic output without options shows things like the user-level CPU usage, system CPU usage, and other details like how long the CPU idled or was interrupted. A high user value can indicate workloads that are CPU-bound, and a high system value could indicate a driver problem or excess system calls. A high wait time (%iowait) can be related to network bottlenecks, and idle times (%idle) can indicate high CPU usage. A high value for hardware interruptions (%irq) can indicate problems with hardware or drivers and a high value for software interruptions (%soft) can indicate networking or input/output processing activity. Having this detailed information is helpful for performance, capacity evaluation, and troubleshooting system issues.

The **ps** command in Linux shows all running instances of a program, i.e., processes (Inhe, 2018). By default, the **ps** command will show only the processes connected to the current window that is run in. The output shows the unique process id (PID), the terminal type that a user is logged into (TTY), the amount of CPU in minutes that a process has been running (TIME), and the command that started the process. To list all system processes use the options -A -e. These options are included in the script for a holistic overview of the system. The **pstree** command displays running processes as a tree structure, which can make it easier to visualize the hierarchy and relationships of system processes (GeeksforGeeks, 2024b). This was also included in the script to gain additional details about how system processes are structured. The last command in the script is **uptime**, and its output shows the current time, how long a system is running, the number of users logged in, and the load time for the prior 1, 5, and 15 minutes (GeeksforGeeks, 2024a).

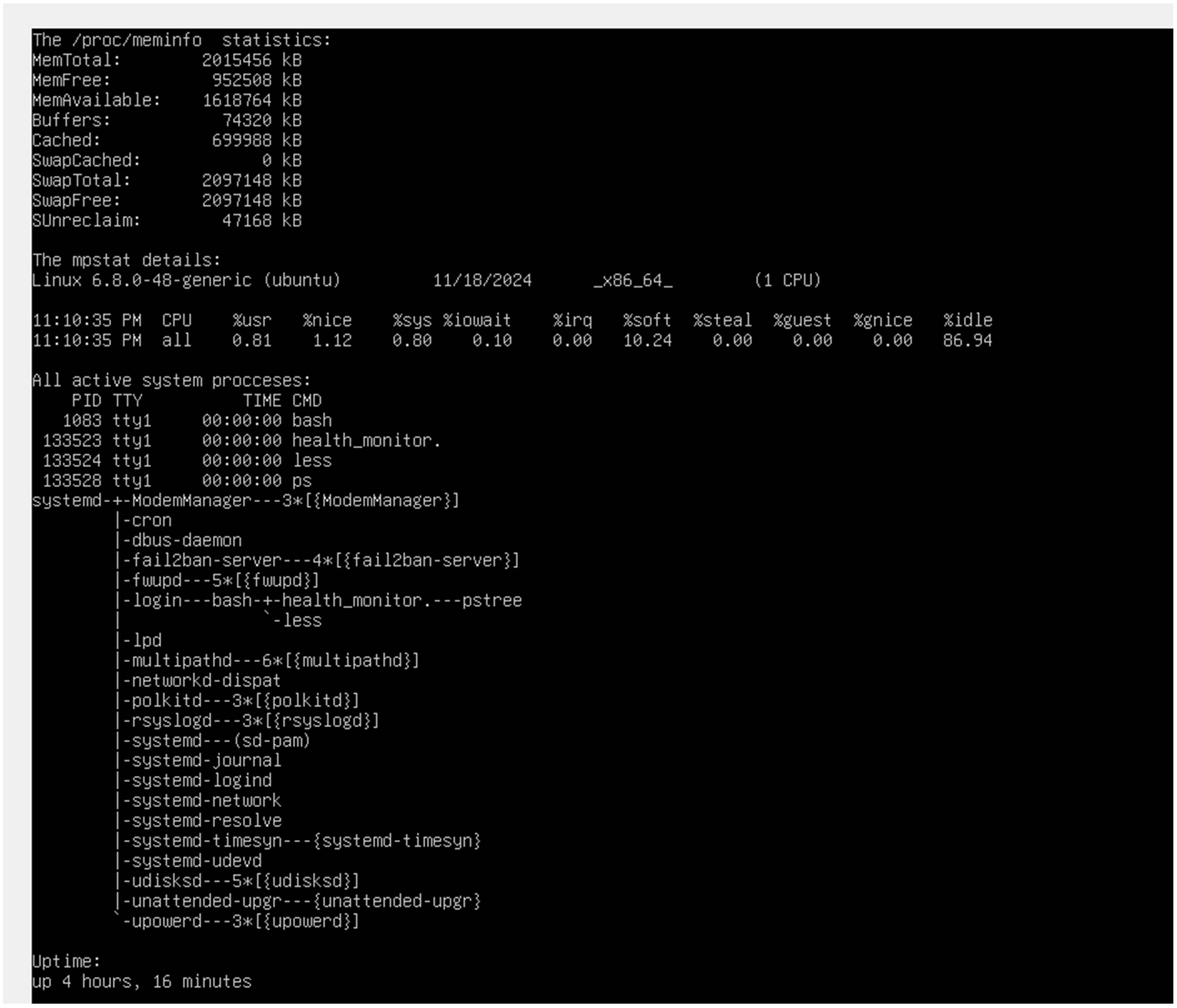
Figure 3.1 Image of health_monitor.sh



```
  GNU nano 7.2                                                      health_monitor.sh
#!/bin/bash

##This script will monitor the health of the server

#This command will show  statistics related to memory, buffers and cache, swap memory, and kernel memory
printf 'The /proc/meminfo  statistics:\n'
cat /proc/meminfo | grep -e "Mem" -e "Buffers" -e "Cached" -e "Swap" -e "reclaim" -e "slab" -e "kernel";


#This command will print an aggregated view of the CPU usage
printf '\nThe mpstat details:\n'
mpstat;

#These commands will look at the processes
printf '\nAll active system procceses:\n'
ps;

pstree;

#This command will show how long the system has been up and running
printf '\nUptime:\n'
uptime -p;

printf '\nDate and time system was last booted:\n'
uptime -s;

printf '\nSystem load average for the last 1, 5, and 15 minutes:\n'
uptime | awk -F'[a-z]:' '{ print $2 }';

now="$(date)"

printf  '\nRun on %(%F)T\n' -1

#print to terminal
 2>&1
```

To run the health_monitor.sh script, navigate to the scripts folder and run the script using the command ./health_monitor.sh. You can also run the script from other directories, just include the full path of the file. For example, from the home directory I could run the script as: ./scripts/health_monitor.sh.

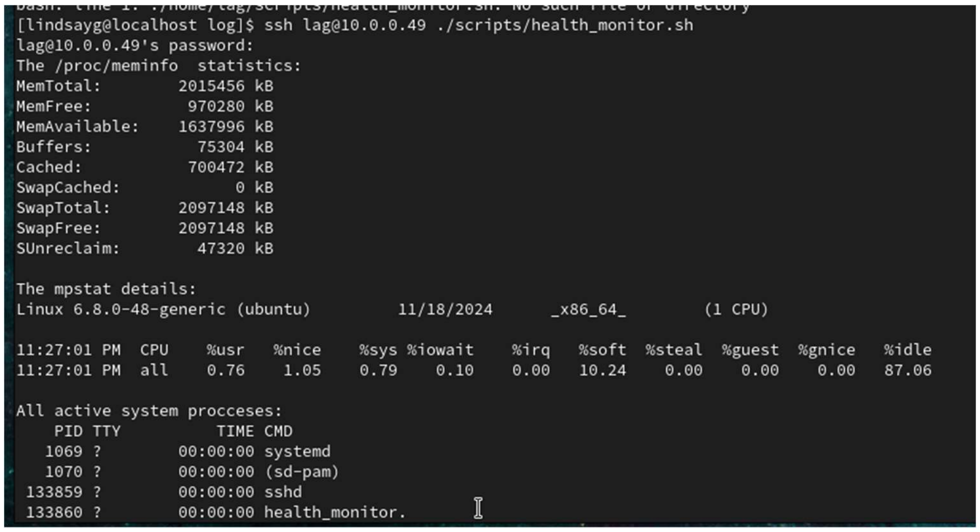
Figure 3.2 Image of partial script output in the Ubuntu terminal



```
The /proc/meminfo  statistics:
MemTotal:        2015456 kB
MemFree:          952508 kB
MemAvailable:    1618764 kB
Buffers:           74320 kB
Cached:           699988 kB
SwapCached:            0 kB
SwapTotal:       2097148 kB
SwapFree:        2097148 kB
SUnreclaim:        47168 kB

The mpstat details:
Linux 6.8.0-48-generic (ubuntu)         11/18/2024      _x86_64_        (1 CPU)

11:10:35 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
11:10:35 PM  all    0.81    1.12    0.80    0.10    0.00   10.24    0.00    0.00    0.00   86.94

All active system procceses:
    PID TTY          TIME CMD
   1083 tty1     00:00:00 bash
 133523 tty1     00:00:00 health_monitor.
 133524 tty1     00:00:00 less
 133528 tty1     00:00:00 ps
systemd-+-ModemManager---3*[{ModemManager}]
        |-cron
        |-dbus-daemon
        |-fail2ban-server---4*[{fail2ban-server}]
        |-fwupd---5*[{fwupd}]
        |-login---bash-+-health_monitor.---pstree
        |              `-less
        |-lpd
        |-multipathd---6*[{multipathd}]
        |-networkd-dispat
        |-polkitd---3*[{polkitd}]
        |-rsyslogd---3*[{rsyslogd}]
        |-systemd---(sd-pam)
        |-systemd-journal
        |-systemd-logind
        |-systemd-network
        |-systemd-resolve
        |-systemd-timesyn---{systemd-timesyn}
        |-systemd-udevd
        |-udisksd---5*[{udisksd}]
        |-unattended-upgr---{unattended-upgr}
        `-upowerd---3*[{upowerd}]

Uptime:
up 4 hours, 16 minutes
```

As an administrator, there may be times when you need to run the script remotely through Secure Socket Shell (SSH). To run the script via SSH use the command syntax **ssh user@host_IP_address ./scripts/health_monitor.sh**. Here is an example of running the script in the Ubuntu server from the CentOS server using SSH.

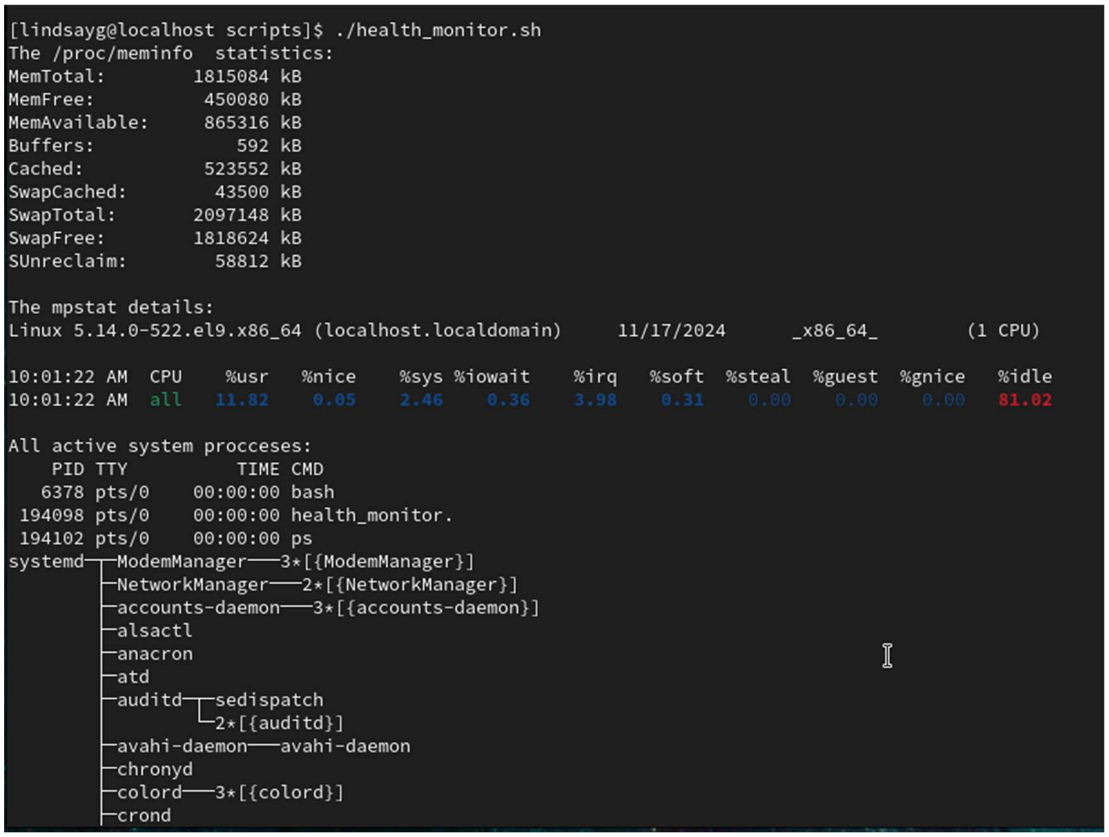
Figure 3.3 Results showing the Ubuntu script successfully running on the CentOS server via SSH



```
[lindsayg@localhost log]$ ssh lag@10.0.0.49 ./scripts/health_monitor.sh
lag@10.0.0.49's password:
The /proc/meminfo  statistics:
MemTotal:        2015456 kB
MemFree:          970280 kB
MemAvailable:    1637996 kB
Buffers:           75304 kB
Cached:           700472 kB
SwapCached:            0 kB
SwapTotal:       2097148 kB
SwapFree:        2097148 kB
SUnreclaim:        47320 kB

The mpstat details:
Linux 6.8.0-48-generic (ubuntu)         11/18/2024      _x86_64_        (1 CPU)

11:27:01 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
11:27:01 PM  all    0.76    1.05    0.79    0.10    0.00   10.24    0.00    0.00    0.00   87.06

All active system procceses:
    PID TTY          TIME CMD
   1069 ?        00:00:00 systemd
   1070 ?        00:00:00 (sd-pam)
 133859 ?        00:00:00 sshd
 133860 ?        00:00:00 health_monitor.
```

The same script works for both servers. The only change that I had to make was to install sysstat to provide the command **mpstat**. When I first ran the script on the CentOS server I got an error that **mpstat** was not found. I tried it in the terminal, and it prompted me to install sysstat. After that, the script ran with no issues.

Figure 3.4 Image of partial script output from health_monitor.sh running in CentOS server



```
[lindsayg@localhost scripts]$ ./health_monitor.sh
The /proc/meminfo  statistics:
MemTotal:        1815084 kB
MemFree:          450080 kB
MemAvailable:     865316 kB
Buffers:             592 kB
Cached:           523552 kB
SwapCached:        43500 kB
SwapTotal:       2097148 kB
SwapFree:        1818624 kB
SUnreclaim:        58812 kB

The mpstat details:
Linux 5.14.0-522.el9.x86_64 (localhost.localdomain)     11/17/2024      _x86_64_        (1 CPU)

10:01:22 AM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
10:01:22 AM  all   11.82    0.05    2.46    0.36    3.98    0.31    0.00    0.00    0.00   81.02

All active system procceses:
    PID TTY          TIME CMD
   6378 pts/0    00:00:00 bash
 194098 pts/0    00:00:00 health_monitor.
 194102 pts/0    00:00:00 ps
systemd─┬─ModemManager───3*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─accounts-daemon───3*[{accounts-daemon}]
        ├─alsactl
        ├─anacron
        ├─atd
        ├─auditd─┬─sedispatch
        │        └─2*[{auditd}]
        ├─avahi-daemon───avahi-daemon
        ├─chronyd
        ├─colord───3*[{colord}]
        ├─crond
```

References

Baeldung. (2024, March 18). *Baeldung*. Baeldung on Linux. https://www.baeldung.com/linux/proc-meminfo

Center for Internet Security. (2016). *CIS Debian Linux 8 Benchmark*. https://web.stanford.edu/~akkornel/CIS_debian8/CIS_Debian_Linux_8_Benchmark_v1.0.0.pdf

CISOfy. (n.d.). *Get Started with Lynis - Installation Guide - CISOfy*. Cisofy.com. https://cisofy.com/documentation/lynis/get-started/

CISOfy. (2013). *Lynis*. Cisofy.com. https://cisofy.com/lynis/

GeeksforGeeks. (2020, May 18). *mpstat Command in Linux with Examples*. GeeksforGeeks. https://www.geeksforgeeks.org/mpstat-command-in-linux-with-examples/

GeeksforGeeks. (2024a, July 15). *Linux | Uptime command with examples - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/linux-uptime-command-with-examples/

GeeksforGeeks. (2024b, October 9). *pstree Command in Linux with Examples*. GeeksforGeeks. https://www.geeksforgeeks.org/pstree-command-in-linux-with-examples/

Hogan, B. (2017, April 28). *How to Perform Security Audits With Lynis on Ubuntu 16.04*. DigitalOcean. https://www.digitalocean.com/community/tutorials/how-to-perform-security-audits-with-lynis-on-ubuntu-16-04

Inhe, M. (2018, April 11). *ps command in Linux with Examples*. GeeksforGeeks. https://www.geeksforgeeks.org/ps-command-in-linux-with-examples/

Mayers, S. (2024, September 23). *2024 NIST Password Guidelines: Enhancing Security Practices for CTO's*. Scytale. https://scytale.ai/resources/2024-nist-password-guidelines-enhancing-security-practices/

Red Hat. (2024). *Chapter 10. Protecting GRUB with a password*. https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/8/html/managing_monitoring_and_updating_the_kernel/assembly_protecting-grub-with-a-password_managing-monitoring-

and-updating-the-kernel#proc_setting-password-protection-only-for-modifying-menu-entries_assembly_protecting-grub-with-a-password

Tutorialspoint. (2022). *Running a Shell Script on a Remote Machine Through SSH*. Tutorialspoint.com. https://www.tutorialspoint.com/running-a-shell-script-on-a-remote-machine-through-ssh