CFC190324
(Linux Fundamentals)
Project Title: Info Extractor
Student Code: S8
Student Name: Jeremiah Lee
Trainer Name: Samson

Table of Contents

Introduction

This report details the development and findings of the "Info Extractor" project. This project aimed to create a script capable of gathering key system information for the purpose of generating comprehensive system reports. The script focuses on extracting crucial data points such as IP address, MAC address, CPU usage, and memory utilisation. This data provides a snapshot of the machine's health and performance, which can be valuable for troubleshooting issues, capacity planning, or security assessments.

Methodologies

In this chapter, we will briefly describe the commands used to obtain the relevant information.

1) Identifying the system's public IP

The following command was used:

```
curl -s ifconfig.io
```

With the following result

```
┌──(kali㉿kali)-[~]
└─$ curl -s ifconfig.io
[redacted].16.154
```

2) Identifying the private IP address assigned to the system's network interface

The following command was used:

```
ifconfig | grep broadcast | awk '{print $2}'
```

With the following result

```
┌──(kali㉿kali)-[~]
└─$ ifconfig | grep broadcast | awk '{print $2}'
192.168.31.130
```

3) Display the MAC address (masking sensitive portions for security)

The following command was used:

```
ip addr | grep 'link/ether' | awk '{print $2}' | cut -c 9-17 | awk '{print "XX:XX:XX:"$0}'
```

With the following results:

```
┌──(kali㉿kali)-[~]
└─$ ip addr | grep 'link/ether' | awk '{print $2}' | cut -c 9-17 | awk '{print "XX:XX:XX:"$0}'
XX:XX:XX::44:a7:e5
```

4) Display the top 5 processes' CPU usage(percentage)

The following command was used:

```
ps aux --sort=-%cpu | head -n6
```

With the following results

```
┌──(kali㉿kali)-[~]
└─$ ps aux --sort=-%cpu | head -n6
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         996  0.7  7.4 482196 150276 tty7     Ssl+ 07:14   1:16 /usr/lib/xorg/Xorg
kali        1403  0.7  2.0 215952 41276 ?        Sl   07:19   1:12 /usr/bin/vmtoolsd -
kali        1342  0.5  1.4 342216 29868 ?        Sl   07:19   0:51 /usr/lib/x86_64-lin
kali        2310  0.2  4.9 460536 100404 ?       Rl   07:20   0:28 /usr/bin/qterminal
kali        1282  0.2  6.2 1273484 125680 ?      Sl   07:19   0:24 xfwm4
```

5) Display the Memory Usage, Free and Used

The following command was used:

```
cat /proc/meminfo | grep Mem
```

With the following result:

```
┌──(kali㉿kali)-[~]
└─$ cat /proc/meminfo | grep Mem
MemTotal:        2015148 kB
MemFree:          828520 kB
MemAvailable:    1236208 kB
```

6) Display the top 10 files (size) from the /home directory

The following command was used:

```
find /home -type f -exec du -b {} + | sort -nr | head -n10
```

With the following result:

```
┌──(kali㉿kali)-[~]
└─$ find /home -type f -exec du -b {} + | sort -nr | head -n10
34539520        /home/kali/Desktop/geany-2.0.tar
32226288        /home/kali/Desktop/geany-2.0/src/.libs/libgeany.so.0.0.0
31295634        /home/kali/Desktop/geany-2.0/scintilla/.libs/libscintilla.a
25222444        /home/kali/Desktop/geany-2.0/scintilla/.libs/liblexilla.a
9552244 /home/kali/Desktop/geany-2.0/src/tagmanager/.libs/libtagmanager.a
9142336 /home/kali/Desktop/geany-2.0/ctags/.libs/libctags.a
9037352 /home/kali/.cache/mozilla/firefox/lcd129e4.default-esr/startupCache/scriptCache.bin
7417591 /home/kali/.cache/mozilla/firefox/lcd129e4.default-esr/startupCache/startupCache.8.little
5242880 /home/kali/.mozilla/firefox/lcd129e4.default-esr/places.sqlite
5242880 /home/kali/.mozilla/firefox/lcd129e4.default-esr/favicons.sqlite
```

7) Display the active system services and status

The following command was used:

```
systemctl list-units --type=service --state=active
```

With the following results

```
┌──(kali㉿kali)-[~]
└─$ systemctl list-units --type=service --state=active
UNIT                                  LOAD   ACTIVE SUB     DESCRIPTION
accounts-daemon.service               loaded active running Accounts Service
colord.service                        loaded active running Manage, Install and Generate Color Profiles
console-setup.service                 loaded active exited  Set console font and keymap
cron.service                          loaded active running Regular background program processing daemon
dbus.service                          loaded active running D-Bus System Message Bus
getty@tty1.service                    loaded active running Getty on tty1
haveged.service                       loaded active running Entropy Daemon based on the HAVEGE algorithm
ifupdown-pre.service                  loaded active exited  Helper to synchronize boot up for ifupdown
keyboard-setup.service                loaded active exited  Set the console keyboard layout
kmod-static-nodes.service             loaded active exited  Create List of Static Device Nodes
lightdm.service                       loaded active running Light Display Manager
```

Overall, the script to extract the information is as follows:

```
#!/bin/bash
#Line 1 Information is a shebang that tells the operating system which interpreter to use in order to execute the script
#Please scroll right to see comments on the commands
echo                                                                                    #Blank line
    echo 'Your Public IP address is'                                                    #Writing String
    curl -s ifconfig.io                                                                 #Command to obtain Public IP
echo                                                                                    #Blank line
    echo 'Your internal IP address is'                                                  #Writing String
    ifconfig | grep broadcast | awk '{print $2}'                                        #Command to obtain Internal IP
echo                                                                                    #Blank line
    echo 'Your MAC address is'                                                          #Writing String
    ip addr | grep 'link/ether' | awk '{print $2}' | cut -c 9-17 | awk '{print "XX:XX:XX:"$0}'  #Command to obtain Mac address and
echo                                                                                    #Blank line
    echo 'Your top 5 processes CPU usage by percentage are'                             #Writing String
    ps aux --sort=-%cpu | head -n6                                                      #Command to obtain Processess
echo                                                                                    #Blank line
    echo 'Here are the details of your memory usage'                                    #Writing String
    cat /proc/meminfo | grep Mem                                                        #Command to obtain System Usage
echo                                                                                    #Blank line
    echo 'Here are your top 10 files from the /home directory'                          #Writing String
    find /home -type f -exec du -b {} + | sort -nr | head -n10                          #Command to obtain Disk Usage
echo                                                                                    #Blank line
    echo 'Display the active system services and status'                                #Writing String
    systemctl list-units --type=service --state=active                                  #Command to obtain active system s
```

Discussion

In this chapter, I will be evaluating the results from the commands used in the previous section. This section will also highlight limitations and challenges faced when using the commands stated previously.

1) Identifying the system's public IP

```
curl -s ifconfig.io
```

.

curl: This is a command-line used to download or retrieve data from URLs.

-s: This flag for curl informs curl to only output the downloaded content itself, without any of the additional information.

ifconfig.io: This is the URL of a web service that provides information about my internet connection.

Limitations:
ifconfig.io is a service provider that provides the public IP address. Like all online services, ifconfig.io may experience downtime or be temporarily unavailable due to numerous reasons like technical issues or high traffic volumes.

2) Identify the private IP address assigned to the system's network interface

```
ifconfig | grep broadcast | awk '{print $2}'
```

Ifconfig: This command is a tool for configuring and displaying information about network interfaces on your system. It interacts with your operating system's network settings, which primarily deal with internal addressing within your local network. When executed without arguments, it gives the following output.

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.31.130  netmask 255.255.255.0  broadcast 192.168.31.255
        inet6 fe80::8733:ee70:bf26:6586  prefixlen 64  scopeid 0x20<link>
        ether [redacted]:44:a7:e5  txqueuelen 1000  (Ethernet)
        RX packets 1062  bytes 144699 (141.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 276  bytes 25000 (24.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

|: Piping, which redirects the output of one command as the input for another command. grep broadcast: grep is used for searching text data, allowing you to narrow down to a specific pattern of text. In this instance “broadcast” was chosen as it was the unique word present in the line which has the internal IP address. Typing ifconfig | grep broadcast gives the following output

```
┌──(kali㉿kali)-[~]
└─$ ifconfig | grep broadcast
        inet 192.168.31.130  netmask 255.255.255.0  broadcast 192.168.31.255
```

awk: filters the given data based on certain criteria.
‘{print $2}’: The curly brackets instructs awk on how to process each line of output. Print informs awk to print something to the output. Whereas $2 refers to the second field in the current line of input. In this case, the numbers “192.168.31.130”, our internal IP address. This is depicted in the following picture, where each “ “ or space separates each field/column.

```
┌──(kali㉿kali)-[~]
└─$ ifconfig | grep broadcast
        inet 192.168.31.130  netmask 255.255.255.0  broadcast 192.168.31.255
```

1 2 3 4 5 6

Which gives the final result

```
┌──(kali㉿kali)-[~]
└─$ ifconfig | grep broadcast | awk '{print $2}'
192.168.31.130
```

3) Display the MAC address (masking sensitive portions for security)

```
ip addr | grep 'link/ether' | awk '{print $2}' | cut -c 9-17 | awk '{print "XX:XX:XX:"$0}'
```

Display the MAC address (masking sensitive portions for security)
Ip addr: Similar to ifconfig, it displays information about my network interfaces. Output is given below

```
┌──(kali㉿kali)-[~]
└─$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether [redacted]:44:a7:e5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.130/24 brd 192.168.31.255 scope global dynamic noprefixroute eth0
       valid_lft 1570sec preferred_lft 1570sec
    inet6 fe80::8733:ee70:bf26:6586/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

|: Piping: which redirects the output of one command as the input for another command.
Grep 'link/ether': as mentioned above, narrows down the output into lines containing the string 'link/ether'.

```
┌──(kali㉿kali)-[~]
└─$ ip addr | grep 'link/ether'
    link/ether [redacted]:44:a7:e5 brd ff:ff:ff:ff:ff:ff
```

awk: filters the given data based on certain criteria
'{print $2}': The curly brackets instructs awk on how to process each line of output. Print informs awk to print something to the output. Whereas $2 refers to the second field in the current line of input. In this case, the output is the unfiltered MAC address. As given below.

```
┌──(kali㉿kali)-[~]
└─$ ip addr | grep 'link/ether' | awk '{print $2}'
[redacted]:44:a7:e5
```

cut -c 9-17: Here the cut command extracts specific characters or sections. In this case, the 9th to 17th position. This results in the output as displayed below.

```
┌──(kali㉿kali)-[~]
└─$ ip addr | grep 'link/ether' | awk '{print $2}' | cut -c 9-17
:44:a7:e5
```

Awk: explained above.
Awk '{print "XX:XX:XX" $0}': This prints the input XX:XX:XX in front of the cut mac address.

```
┌──(kali㉿kali)-[~]
└─$ ip addr | grep 'link/ether' | awk '{print $2}' | cut -c 9-17 | awk '{print "XX:XX:XX"$0}'
XX:XX:XX:44:a7:e5
```

Limitations

A) Permissions

In some Linux systems, ip addr might require root privileges to access all network interface details.

B) Interface identification

Information is displayed for all network interfaces in the system. If one has multiple interfaces like Wi-fi and Ethernet, one needs to identify the correct one based on its name to find the correct MAC address.

4) Display the top 5 processes' CPU usage(percentage)

```
ps aux --sort=-%cpu | head -n6
```

ps:The process status command used to view information about running processes
a: Shows all processes, including those of other users
u: shows detailed information about each process, including the user who owns it, and the percentage of CP its using among other things.
x: includes processes that don't have controlling terminal (daemon processes/background processes)
--sort=-%cpu: This sorts the output based on the CPU usage (%cpu). The "-" sign indicates sorting by descending order with the highest CPU usage listed first.
|: piping as explained previously
head -n6: This display the top 6 rows in the output. Head -n6 is chosen instead of head-n5 as the headers as also included in the output.

```
┌──(kali㉿kali)-[~]
└─$ ps aux --sort=-%cpu | head -n6
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         996  0.7  7.4 482196 150276 tty7     Ssl+ 07:14   1:16 /usr/lib/xorg/Xorg
kali        1403  0.7  2.0 215952 41276 ?        Sl   07:19   1:12 /usr/bin/vmtoolsd -
kali        1342  0.5  1.4 342216 29868 ?        Sl   07:19   0:51 /usr/lib/x86_64-lin
kali        2310  0.2  4.9 460536 100404 ?       Rl   07:20   0:28 /usr/bin/qterminal
kali        1282  0.2  6.2 1273484 125680 ?      Sl   07:19   0:24 xfwm4
```

In a cybersecurity context, a high amount of CPU usage can be linked to underlying security threats like Malware, Spyware or even trojans. Such threats run taxing processes in the background contributing to high CPU usage (ReasonLabs, no date).

Limitations:
Ps aux provides only a snapshot of cpu usage at the time the command is executed. In reality, the actual CPU usage dynamically changes overtime as processes change their CPU consumption.

5) Display the Memory Usage, Free and Used

```
cat /proc/meminfo | grep Mem
```

cat: This command is used to display the contents of a file
/proc/meminfo: This is a special file in the /proc pseudo-filesystem which provides information about the system's memory usage
|: Explained previously
grep Mem: narrows down output to the case sensitive string "Mem".

```
┌──(kali㉿kali)-[/proc]
└─$ cat /proc/meminfo | grep Mem
MemTotal:        2015124 kB
MemFree:          796492 kB
MemAvailable:    1176968 kB
```

Case sensitivity is important as using grep with "mem" gives the following information

```
┌──(kali㉿kali)-[/proc]
└─$ cat /proc/meminfo | grep mem
Shmem:             20488 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
```

Having quick access to a system's memory plays an important role in memory forensics for incident reporting. By capturing the memory of the device quickly, we are able to identify potential malware (Neil, 2022).

Limitations
Similar to the previous section. The command used only provides a snapshot of memory usage of the machine. It also does not provide detailed history of memory usage in the machine.

6) Display the top 10 files (size) from the /home directory

```
find /home -type f -exec du -b {} + | sort -nr | head -n10
```

find: This command searches for files within a directory
/home: This specifies the directory to search in (in this case /home)
-type f: This option tells find to only search for files (not directories)

-exec: This option allows executing a command based on the files find discovers
du -b {} +: Here, du -b calculates the size of each file in bytes (-b flag), and {} represents a placeholder for each found file. The + tells du to process all found files at once.
|: Piping explained earlier
sort: Command sorts the piped data (in this case via file size)
-n: this option sorts based on numeric values
-r: This option sorts in reverse order (from big to smallest)
|: piping
head -n10: Command displays the beginning of the piped data, with -n10 flag specifying the number of lines to display.

```
┌──(kali㉿kali)-[~]
└─$ find /home -type f -exec du -b {} + | sort -nr | head -n10
34539520        /home/kali/Desktop/geany-2.0.tar
32226288        /home/kali/Desktop/geany-2.0/src/.libs/libgeany.so.0.0.0
31295634        /home/kali/Desktop/geany-2.0/scintilla/.libs/libscintilla.a
25222444        /home/kali/Desktop/geany-2.0/scintilla/.libs/liblexilla.a
9552244 /home/kali/Desktop/geany-2.0/src/tagmanager/.libs/libtagmanager.a
9142336 /home/kali/Desktop/geany-2.0/ctags/.libs/libctags.a
9037352 /home/kali/.cache/mozilla/firefox/lcd129e4.default-esr/startupCache/scriptCache.bin
7417591 /home/kali/.cache/mozilla/firefox/lcd129e4.default-esr/startupCache/startupCache.8.little
5242880 /home/kali/.mozilla/firefox/lcd129e4.default-esr/places.sqlite
5242880 /home/kali/.mozilla/firefox/lcd129e4.default-esr/favicons.sqlite
```

Limitations

The find command might not have access to all directories within /home if the user running the command lacks permissions.
In an example, the file is moved to another user's folder

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo mv geany-2.0.tar /home/jeremiah/geany-2.0.tar
```

The following error is shown as we are unable to access the folder.

```
┌──(kali㉿kali)-[~]
└─$ find /home -type f -exec du -b {} + | sort -nr | head -n10
find: '/home/jeremiah': Permission denied
32226288        /home/kali/Desktop/geany-2.0/src/.libs/libgeany.so.0.0.0
31295634        /home/kali/Desktop/geany-2.0/scintilla/.libs/libscintilla.a
25222444        /home/kali/Desktop/geany-2.0/scintilla/.libs/liblexilla.a
9552244 /home/kali/Desktop/geany-2.0/src/tagmanager/.libs/libtagmanager.a
9142336 /home/kali/Desktop/geany-2.0/ctags/.libs/libctags.a
9037352 /home/kali/.cache/mozilla/firefox/lcd129e4.default-esr/startupCache/scriptCache.bin
7417591 /home/kali/.cache/mozilla/firefox/lcd129e4.default-esr/startupCache/startupCache.8.little
5242880 /home/kali/.mozilla/firefox/lcd129e4.default-esr/places.sqlite
5242880 /home/kali/.mozilla/firefox/lcd129e4.default-esr/favicons.sqlite
4178552 /home/kali/Desktop/geany-2.0/scintilla/src/.libs/libscintilla_la-Editor.o
```

7) Display the active system services and status

```
systemctl list-units --type=service --state=active
```

systemctl: This is the program name used to interact with systemd, the system service manager used by most modern Linux distributions

list-units: The subcommand of systemctl that instructs it to list information about systemd units. A unit can be a service, a device, a mount point, a target, or another type of system entity managed by systemd

--type=service: This is an option that filters the output to only show units of the type "service". Services are programs that run in the background and perform specific tasks on the system.

--state=active: This is another option that further filters the output to only show services that are currently in the "active" state. An active service is a service that is currently running.

In summary, this command lists all units, filters the list to show only units of type “service” and further filters to show only those in the active status.

```
┌──(kali㉿kali)-[~]
└─$ systemctl list-units --type=service --state=active
  UNIT                                  LOAD   ACTIVE SUB     DESCRIPTION
  accounts-daemon.service               loaded active running Accounts Service
  blueman-mechanism.service             loaded active running Bluetooth management mechanism
  colord.service                        loaded active running Manage, Install and Generate Color Profiles
  console-setup.service                 loaded active exited  Set console font and keymap
  cron.service                          loaded active running Regular background program processing daemon
  dbus.service                          loaded active running D-Bus System Message Bus
  getty@tty1.service                    loaded active running Getty on tty1
  haveged.service                       loaded active running Entropy Daemon based on the HAVEGE algorithm
  ifupdown-pre.service                  loaded active exited  Helper to synchronize boot up for ifupdown
  keyboard-setup.service                loaded active exited  Set the console keyboard layout
```

Conclusion

Despite achieving the objective of gathering required information, there were potential limitations faced upon further investigation of the script. Some of the issues were due to the user or the command's lack of authorisation permissions, while others involved the static nature of the command itself. The following chapter will address said issues and provide recommendations for improvement.

Recommendations

1) Limitation of service providers of external service providers

As previously mentioned, ifconfig.io is a service provider that could experience downtime or be temporarily unavailable due to numerous reasons like technical issues or high traffic volumes. Alternative service providers include ifconfig.co, ifconfig.me, ifconfig.il and icanhazip.com.

2) Limitation of interface identification when obtaining MAC address.

The script written assumes that the machine only has one network interface. In some instances where there are two separate network interfaces, (Wired Ethernet Adapter and Wi-Fi) there will be 2 instances of 'link/ether' that may be displayed, which may lead to inaccurate information regarding the MAC address.

In such circumstances, if a specific MAC address (Ethernet or Wifi) is known, the following command can be used to replace ip addr.

```
┌──(kali㉿kali)-[~]
└─$ ip addr show eth0
```

Ip addr show eth0 can be used for wired ethernet adapter, whereas wlan0 can be used for Wi-Fi.

The full result command would be as follows:

```
┌──(kali㉿kali)-[~]
└─$ ip addr show eth0 | grep link/ether | awk '{print $2}' | cut -c 9-17 | awk '{print "XX:XX:XX:"$0}'
XX:XX:XX::44:a7:e5
```

3) Limitations of "ps" and "cat /proc/meminfo" in obtaining dynamic CPU and memory usage.

The commands "ps" and "cat /proc/meminfo" only provide a snapshot of the respective information. In reality, CPU and memory usage is constantly changing in a machine. Hence in such cases, the command "top" can be used to provide more information.

“Top” command when used automatically sorts with the highest %CPU being used

```
top - 06:38:32 up 25 min,  1 user,  load average: 0.07, 0.12, 0.09
Tasks: 202 total,   1 running, 201 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.1 us,  0.2 sy,  0.0 ni, 99.6 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :   1967.9 total,    787.4 free,    720.3 used,    608.6 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1247.6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    606 root      20   0  242120  12788   7552 S   0.3   0.6   0:01.56 vmtoolsd
    970 root      20   0  373104  94492  52380 S   0.3   4.7   0:03.47 Xorg
   1302 kali      20   0  342216  28116  20804 S   0.3   1.4   0:03.40 panel-15-genmon
   1353 kali      20   0  216492  41540  30260 S   0.3   2.1   0:01.91 vmtoolsd
   9404 kali      20   0   11740   5760   3584 R   0.3   0.3   0:01.44 top
  11830 root      20   0       0      0      0 I   0.3   0.0   0:00.30 kworker/0:0-events
```

Pressing “M” will sort by the highest memory usage.

```
top - 06:38:11 up 25 min,  1 user,  load average: 0.10, 0.13, 0.09
Tasks: 202 total,   1 running, 201 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1967.9 total,    791.7 free,    716.8 used,    606.9 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1251.1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1242 kali      20   0 1273480 124184  77816 S   0.0   6.2   0:01.25 xfwm4
   1626 kali      20   0  456880  97888  82836 S   0.0   4.9   0:01.43 qterminal
    970 root      20   0  373104  94492  52380 S   0.0   4.7   0:03.42 Xorg
   1293 kali      20   0  483640  62584  36908 S   0.0   3.1   0:00.74 xfdesktop
   1341 kali      20   0  449656  54036  30720 S   0.0   2.7   0:00.43 blueman-applet
   1376 kali      20   0  625480  50588  37592 S   0.0   2.5   0:00.16 nm-applet
   1297 kali      20   0  466096  48172  34400 S   0.0   2.4   0:00.29 panel-1-whisker
   1282 kali      20   0  467156  47068  34704 S   0.0   2.3   0:00.39 xfce4-panel
   1305 kali      20   0  391692  44800  32668 S   0.0   2.2   0:00.27 panel-18-power-
   1304 kali      20   0  464868  44352  32304 S   0.0   2.2   0:00.14 panel-17-notifi
   1303 kali      20   0  462076  43332  34900 S   0.0   2.2   0:00.18 panel-16-pulsea
```

4) Unable to access the top 10 files in /home directory

As previously mentioned, after moving the file “geany-2.0.tar” from kali to another user’s folder, the command was unable to identify that it was the largest file in the /home directory because it did not have the necessary permissions to view all the files in the /home directory. Hence, adding “sudo” into the front of the command allows it to have a higher level of permissions to obtain access to the files.

```
┌──(kali㉿kali)-[~]
└─$ sudo find /home -type f -exec du -b {} + | sort -nr | head -n10
34539520        /home/jeremiah/geany-2.0.tar
32226288        /home/kali/Desktop/geany-2.0/src/.libs/libgeany.so.0.0.0
31295634        /home/kali/Desktop/geany-2.0/scintilla/.libs/libscintilla.a
25222444        /home/kali/Desktop/geany-2.0/scintilla/.libs/liblexilla.a
9552244 /home/kali/Desktop/geany-2.0/src/tagmanager/.libs/libtagmanager.a
9142336 /home/kali/Desktop/geany-2.0/ctags/.libs/libctags.a
9037352 /home/kali/.cache/mozilla/firefox/lcd129e4.default-esr/startupCache/scriptCache.bin
7417591 /home/kali/.cache/mozilla/firefox/lcd129e4.default-esr/startupCache/startupCache.8.little
5242880 /home/kali/.mozilla/firefox/lcd129e4.default-esr/places.sqlite
5242880 /home/kali/.mozilla/firefox/lcd129e4.default-esr/favicons.sqlite
```

References

Fox, N. (2022) *Memory forensics for incident response*, *Varonis*. Available at: https://www.varonis.com/blog/memory-forensics (Accessed: 14 April 2024).

ReasonLabs (no date) *What is high CPU usage?*, *What is High CPU Usage? The Significance of CPU Usage in Cybersecurity*. Available at: https://cyberpedia.reasonlabs.com/EN/high%20cpu%20usage.html (Accessed: 14 April 2024).