Collecting Hostname, Date, and Time

- Identify the computer name using the **hostname** command

  Command:

  `hostname`

- This command can be useful while examining logs and network traffic

- Check the **date and time** of the machine to build a proper timeline of events

  Command:

  `date`

  `cat /etc/timezone`

```
root@ubuntu: ~
File  Edit  View  Search  Terminal  Help
root@ubuntu:~# hostname
ubuntu
root@ubuntu:~#
```

```
root@ubuntu: /home/investigator
File  Edit  View  Search  Terminal  Help
root@ubuntu:/home/investigator# date
Wed Apr 22 23:53:45 PDT 2020
root@ubuntu:/home/investigator# cat /etc/timezone
America/Los_Angeles
root@ubuntu:/home/investigator#
```

Collecting Hostname, Date, and Time (Cont'd)

- Alternately, you can **calculate the epoch time** (count of the number of seconds from the Unix OS starting point) of the system and convert it w.r.t your time zone

  Command:

  `date +%s`

  Note: The Unix epoch timestamp begins on 1st January 1970 00:00:00 UTC (in seconds), whereas the epoch timestamps for HFS+ and Cocoa in Apple begin on 1st January 1904 00:00:00 UTC (in seconds) and 1st January 2001 00:00:00 UTC (in seconds), respectively.

```
root@james-Virtual-Machine: /home/james
root@james-Virtual-Machine:/home/james# date +%s
1598025566
root@james-Virtual-Machine:/home/james#
```

- Upon obtaining the epoch timestamp, you can use online or offline converters to convert the epoch time to original time. Here, we are doing the conversion in *www.epochconverter.com*

Epoch Converter - Unix Timestamp Converter - Mozilla Firefox

https://www.epochconverter.com

EpochConverter

Epoch & Unix Timestamp Conversion Tools

The current Unix epoch time is 1598026162

Convert epoch to human-readable date and vice versa

1598025566 | Timestamp to Human date | reset

Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.

Assuming that this timestamp is in **seconds**:

GMT	: Friday, August 21, 2020 3:59:26 PM
Your time zone	: Friday, August 21, 2020 11:59:26 AM GMT-04:00 DST
Relative	: 3 minutes ago

Yr	Mon	Day	Hr	Min	Sec			
2020	8	21	4	2	10	PM	GMT	Human date to Timestamp

This website uses cookies to ensure you get the best experience on our website. Learn more | Got it!

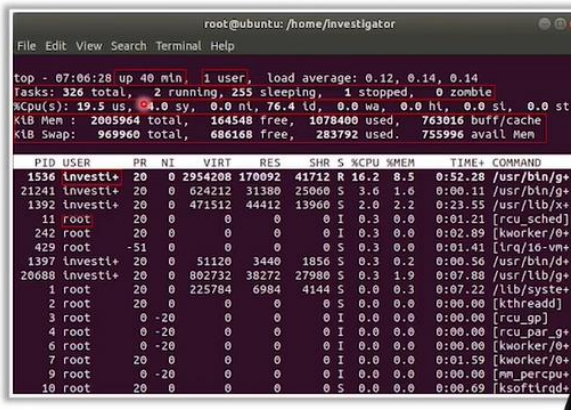
Collecting Uptime Data

- The **uptime** command in Linux system displays how long the system has been running since the last restart

- This command also returns the current time, number of presently logged-in users, system load averages, etc.

Command:

`uptime`



Collecting Network Information

- The following syntax displays all **Network Interface Controllers** (NICs) and associated IP addresses associated with them

Syntax:

`ip addr show`

Note:

- lo, ens33 are NICs
- State **UNKNOWN – NIC** is operational but there is no connection
- State **UP – NIC** is operational and there is a connection

Collecting Network Information (Cont'd)

The **netstat** command can be used to extract network information

It displays network connections, routing tables, and a number of network interface (network interface controller or software-defined network interface) and network protocol statistics

```
root@ubuntu: /home/investigator
File Edit View Search Terminal Help
root@ubuntu:/home/investigator# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    55977    /run/user/0/systemd/notify
unix  2      [ ]         DGRAM                    45900    /run/user/1000/systemd/notify
unix  2      [ ]         DGRAM                    38957    /run/user/121/systemd/notify
unix  25     [ ]         DGRAM                    24155    /run/systemd/journal/dev-log
unix  2      [ ]         DGRAM                    24157    /run/systemd/journal/syslog
unix  3      [ ]         DGRAM                    23991    /run/systemd/notify
unix  10     [ ]         DGRAM                    24003    /run/systemd/journal/socket
unix  3      [ ]         STREAM     CONNECTED     48776    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     47421    @/tmp/.ICE-unix/1432
unix  3      [ ]         STREAM     CONNECTED     46637    @/tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     35744    /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     47539    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     46934    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     44615
unix  3      [ ]         STREAM     CONNECTED     35741    /var/run/dbus/system_bus_socket
unix  3      [ ]         DGRAM                    32976
unix  3      [ ]         STREAM     CONNECTED     63601    /run/user/1000/pulse/native
unix  3      [ ]         STREAM     CONNECTED     47491    /run/systemd/journal/stdout
```

```
root@ubuntu: /home/investigator
File Edit View Search Terminal Help
root@ubuntu:/home/investigator# netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
ens33     1500    55705      0      0 0         30553      0      0      0 BMRU
lo       65536    18197      0      0 0         18197      0      0      0 LRU
root@ubuntu:/home/investigator#
```

To view list of **network interfaces** on the system

Command:

`netstat -i`

Viewing Network Routing Tables

- The **netstat** command can also be used to print **routing tables**

  Command:

  `netstat -rn`

  -r displays the kernel IP routing table

  -n displays the numerical addresses

```
root@ubuntu: /home/investigator
File Edit View Search Terminal Help
root@ubuntu:/home/investigator# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.135.2   0.0.0.0         UG        0 0          0 ens33
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 ens33
192.168.135.0   0.0.0.0         255.255.255.0   U         0 0          0 ens33
root@ubuntu:/home/investigator#
```

```
root@ubuntu: /home/investigator
File Edit View Search Terminal Help
root@ubuntu:/home/investigator# ip r
default via 192.168.135.2 dev ens33 proto dhcp metric 100
169.254.0.0/16 dev ens33 scope link metric 1000
192.168.135.0/24 dev ens33 proto kernel scope link src 192.168.135.136 metric 100

root@ubuntu:/home/investigator#
```

- In Linux, the **routing table** provides information on the process of forwarding TCP/IP data packets

  Command:

  `ip r`

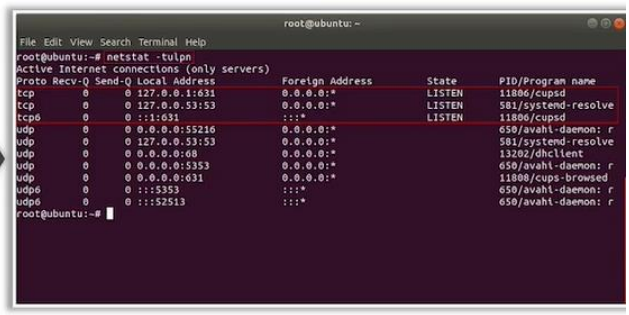
Finding Programs/Processes Associated with a Port

- To detect **intrusions**, it is necessary to collect **open port** information
- It is also important to check if there are any **programs/processes** associated with **open ports**

  Command:

  `netstat -tulpn`

- In the screenshot, **cupsd** is the process with PID 11806, running on port 631



- Another command to list the **processes running** on **open ports**

  Command:

  `lsof -i -P -n | grep LISTEN`

- The **grep command** is used to filter ports in the LISTEN state

Collecting Data on Open Files

- You can run `lsof` command to list all open files as well as the active processes that opened them on the system

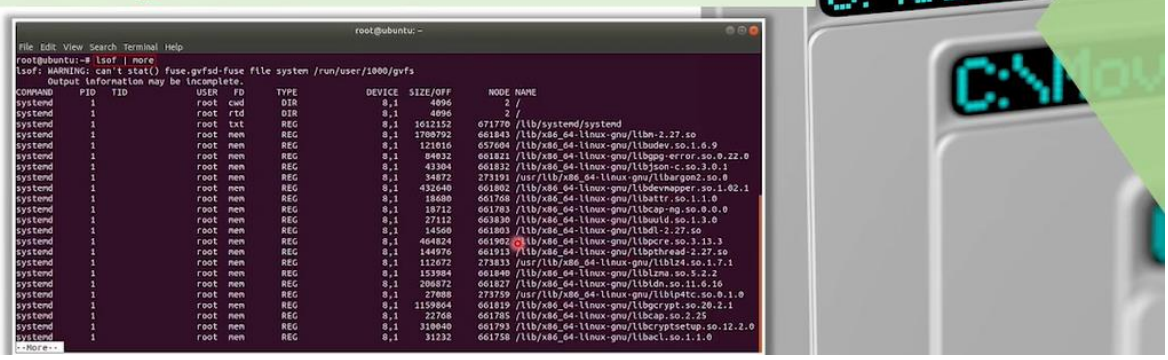
  Command:

  `lsof`

- To list the open files for the user currently logged into the system

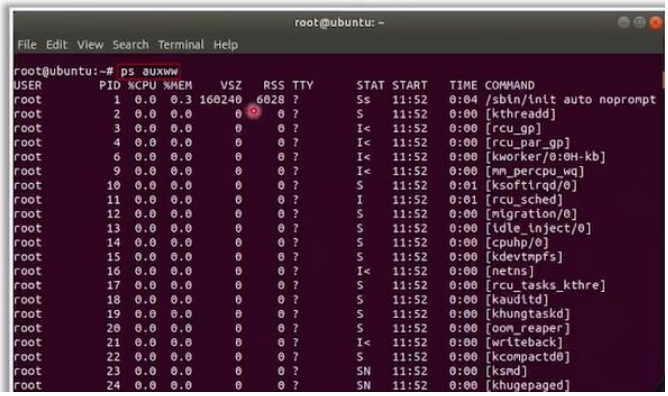
  Command:

  `lsof -u <user_name>`



Viewing Running Processes in the System

- Run the **ps** command to view the processes running on the system
- It provides a **snapshot** of the **current processes** along with detailed information, such as the **user id**, **CPU usage**, **memory usage**, and **command name**
- Check the **process tree** to determine any suspicious **child processes** and **dependencies**



Collecting Kernel Information

- Use the following **commands** to check the Linux **kernel version** on a system:

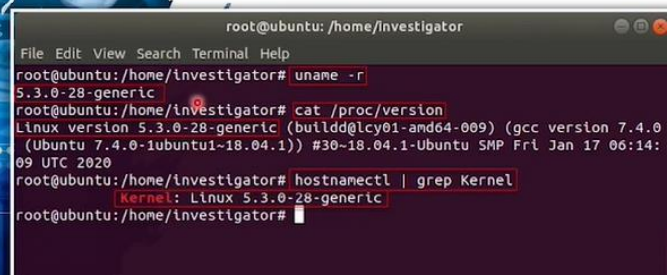
  `uname -r`

  (or)

  `cat /proc/version`

  (or)

  `hostnamectl | grep Kernel`



Collecting User Account Information

The **/etc/passwd** file running on a Linux system stores **local user** account information

Analyzing the /etc/passwd file allows the investigator to view **the user accounts** on the system

Command:

```
cat /etc/passwd
```

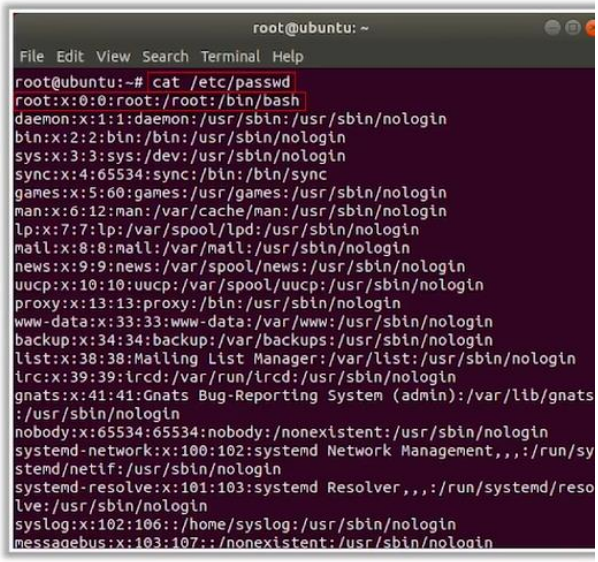
Command given to list only **usernames** in the output

```
cut -d: -f1 /etc/passwd
```

Each line in the output represents the **login information** of a single **user** and includes seven fields separated by colon (:)

You can observe the following about the output format of the **/etc/passwd** file by analyzing the first entry in the screenshot:

root – Username
x – Password ('x' denotes encrypted)
0 – User ID ('0' is reserved for root)
0 – Group ID
root – User ID information
/root – Home directory
/bin/bash – Absolute path to the user's login shell



Collecting Currently Logged-in Users and Login History Information (Cont'd)

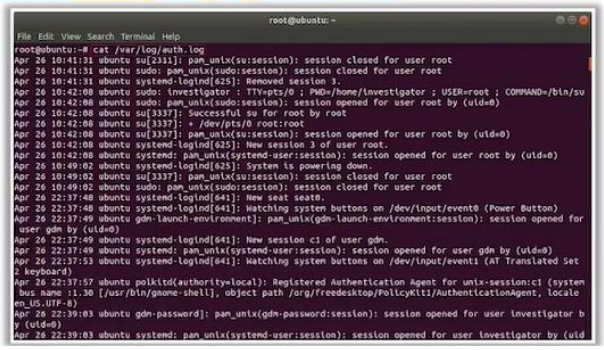
- The **/var/log/auth.log** file logs information related to the user's authentication and authorization events, user remote logins, sudo logins, SSH logins, etc.

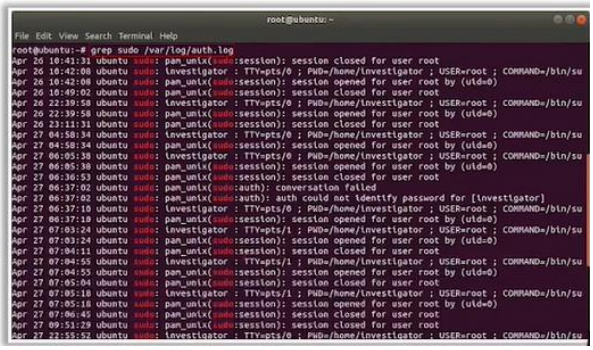
  Command:

  `cat /var/log/auth.log`

- The following command filters out **sudo** commands

  `grep sudo /var/log/auth.log`

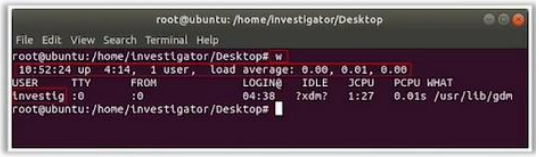




- To find the currently **logged in user** in the system

  Command:

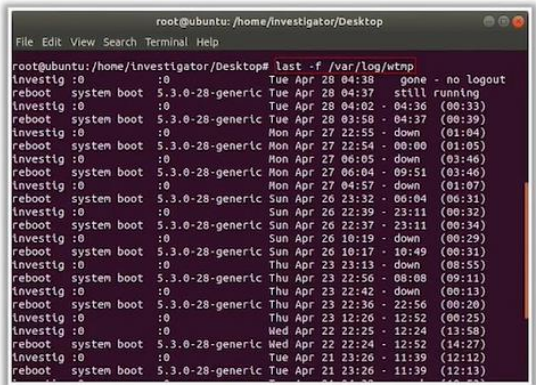
  `w`



- The log file **/var/log/wtmp** maintains information about the user login history, system reboot time and system status
- The **last** command pulls the login history from the **wtmp** log file

  Command:

  `last -f /var/log/wtmp`



Collecting System Logs Data

- On a Linux machine, the system logs are located in the directory **/var/log/syslog**
- The **syslog configuration file** stores system messages from logging facility and collects data logs of various programs and services, including the kernel

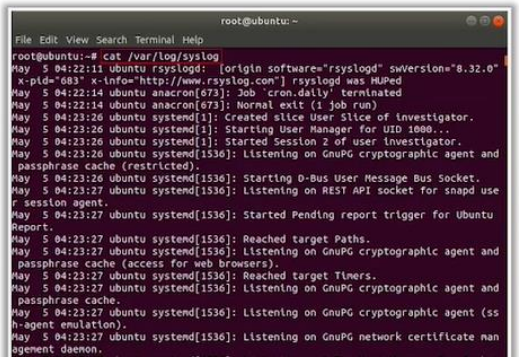
Command:

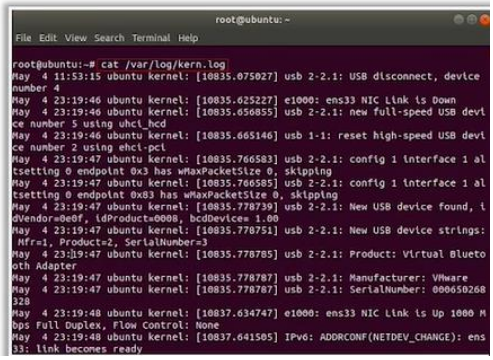
```
cat /var/log/syslog
```

- Analyzing **Linux kernel logs** located at **/var/log/kern.log** can be helpful for troubleshooting custom kernels

Command:

```
cat /var/log/kern.log
```





Linux Log Files

Log Location	Content Description
/var/log/auth.log	System authorization information, including user logins and authentication mechanism
/var/log/kern.log	Initialization of kernels, kernel errors or informational messages sent from the kernel
/var/log/faillog	Failed user login attempts
/var/log/lpr.log	Printer logs
/var/log/mail.*	All mail server message logs
/var/log/mysql.*	All MySQL server logs
/var/log/apache2/*	All Apache web server logs
/var/log/apport.log	Application crash report/log
/var/log/lighttpd/*	Lighttpd web server log files directory
/var/log/daemon.log	Running services, such as squid and ntpd
/var/log/debug	Debugging log messages
/var/log/dpkg.log	Package installation or removal logs

File System Analysis Using The Sleuth Kit: fsstat

- In Linux systems, the **fsstat** command provides information associated with the given file system
- The output of this command is filesystem-specific and consists of several information such as the file system type, volume ID, last mounted timestamps and last mounted directory

Command:

```
fsstat -i <input _filetype>
<filename.extension>
```

```
root@ubuntu: /home/investigator/Desktop
File  Edit  View  Search  Terminal  Help
root@ubuntu:/home/investigator/Desktop# fsstat -i raw Evidence.dd
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext4
Volume Name:
Volume ID: c322aba24d160eab914b7ee5f9917521

Last Written at: 2020-05-01 08:14:49 (PDT)
Last Checked at: 2020-04-23 23:40:05 (PDT)

Last Mounted at: 2020-05-01 08:12:00 (PDT)
Unmounted properly
Last mounted on: /mnt/ext4

Source OS: Linux
Dynamic Structure
Compat Features: Journal, Ext Attributes, Resize Inode, Dir Index
InCompat Features: Filetype, Extents, 64bit, Flexible Block Groups,
Read Only Compat Features: Sparse Super, Large File, Huge File, Extra Inode Size

Journal ID: 00
```

Or Autopsy using for file system analysis

Memory Forensics: Introduction

1. Memory forensics involves **forensic analysis of RAM dumps** captured from a running machine
2. Forensic analysis of **RAM dump** provides insights into processes running in the memory, network information, unauthorized access to the system, loaded modules, recently executed commands, injected code fragments, etc.
3. Such information can help the investigator **uncover malware attacks** or any other malicious behavior that has occurred on the target machine

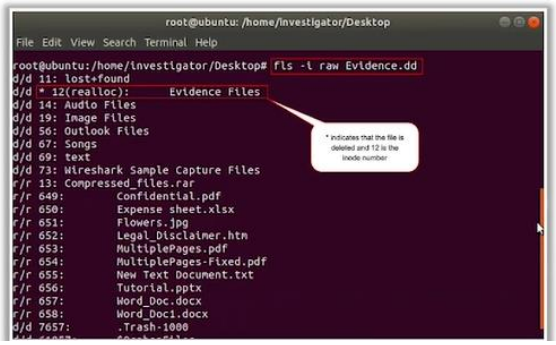
Note: The investigator should procced with the forensic examination based on the **information/events recorded** by the incident response team

File System Analysis Using The Sleuth Kit: fls and istat

- Run the **fls** command to list the files and directories available in an image file
- This command is also useful to view **recently deleted files**

Command:

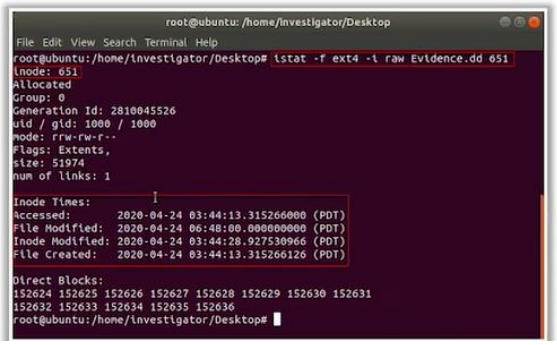
```
fls -i <image_type> <imagefile_name>
```



- Use **istat** command that displays the metadata of a file, such as MAC times, file size, and file access permissions, by specifying a particular inode number

Command:

```
istat -f <fstype> -i <imgtype>
<imagefile_name> <inode_number>
```



Malware Analysis Using Volatility Framework

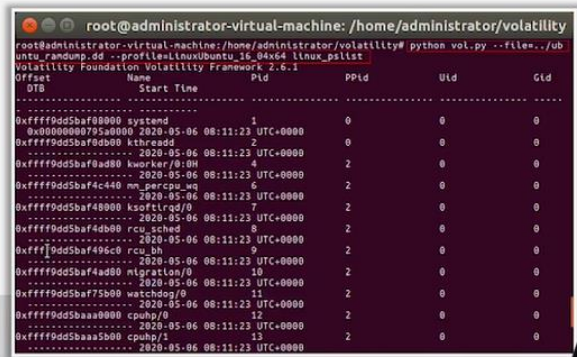
- After acquiring RAM dumps from the target machine, the investigator should analyze those dumps using tools such as **Volatility** to **identify** the occurrence of **malicious activity**
- To examine memory dumps using Volatility Framework, the investigator should **create a Linux profile** that matches the kernel version of the target RAM dump (which is used for analysis)

The **pslist** plugin lists all the **processes** that were running on the machine when the memory dump was captured

Command:

```
python vol.py --file=<file_name> --
profile=<Linux_profile_name> linux_pslist
```

Note: In this case, the Linux profile is ***Linux Ubuntu_16.04 x64***

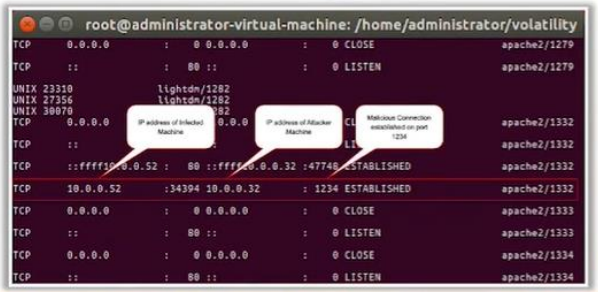


Malware Analysis Using Volatility Framework (Cont'd)

- Use the **netstat** plugin to search for malicious network communication on the machine

Command:

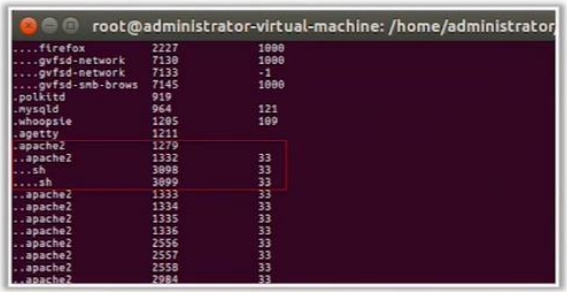
```
python vol.py --file=<file_name> --
profile=<Linux_profile_name>
linux_netstat
```



- The **pstree** plugin displays the parent and associated child processes generated using a malicious backdoor
- From the screenshot below, it can be observed that the **apache2** process with **PID 1279** started another **apache2** process with **PID 1332**
- This indicates that the **process** with **PID 1332** is establishing malicious communication

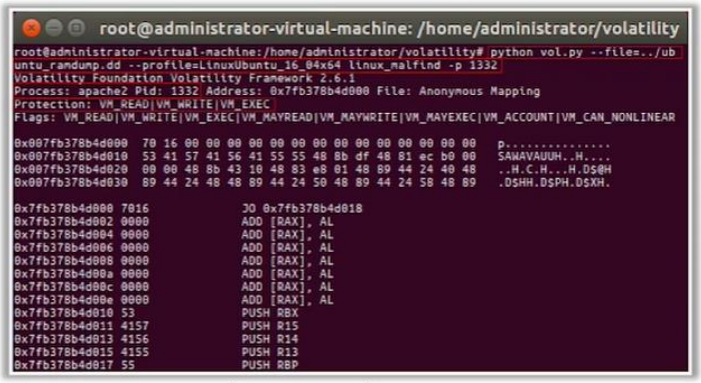
Command:

```
python vol.py --file=<file_name> --
profile=<Linux_profile_name> linux_pstree
```



Malware Analysis Using Volatility Framework (Cont'd)

- The **malfind** plugin helps the investigator identify any **remote/hidden code injections** in the memory
  - **Command:**
  - `python vol.py --file=<file_name> --profile=<Linux_profile_name> linux_malfind`
- From **pstree** output, the process with **PID 1332** is identified as **malicious**. You can utilize **malfind** plugin to check whether **PID 1332** is a legitimate process.
- When **malfind** plugin is run with PID 1332, the parameter '**Protection**' shows that the process is marked with **Read, Write** and **Execute** permissions. This indicates that some **malicious code** has been **injected** into the process.



Carving Memory Dumps Using PhotoRec Tool

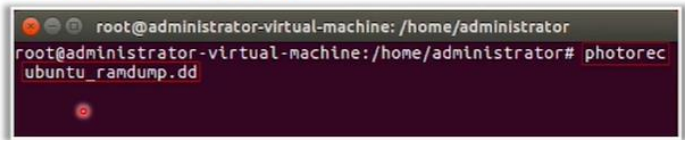
PhotoRec is an open-source tool that uses data carving techniques to **recover deleted files/lost data** from a drive or an image file

Memory dumps **contain volatile data** pertaining to logged on users, shared files, recently accessed media files and chats via social networks, accessed webpages, etc.

Identifying and extracting these files allows the forensic investigators to perform a more detailed investigation

Carving data from the memory dump

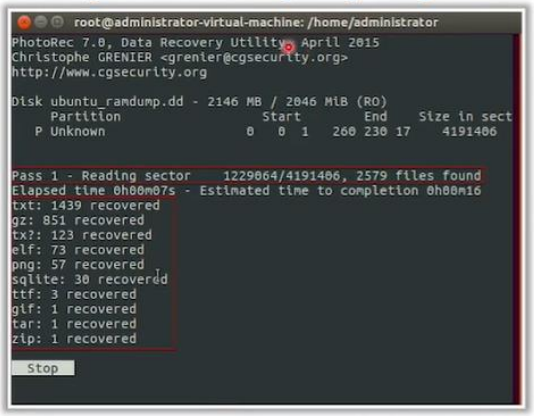
- Run the **PhotoRec** tool and execute the below command
  Command:
  `photorec <Imagefile_name>`



Carving Memory Dumps Using PhotoRec Tool (Cont'd)

- Use anti-malware tools to **scan** the data extracted from the memory dumps for viruses
- This enables the detection of **any malicious data** in the memory dumps that could be helpful during an investigation

Extracting data from memory dumps using PhotoRec



Data recovered from the image file