Report on Info Extractor

Introduction

This report presents the findings of an analysis conducted on a computer system's network infrastructure by running a bash shell script. The purpose of the analysis was to evaluate the system's connectivity and performance, focusing on its public and private IP addresses, MAC address, CPU usage, memory usage, active system services, and largest files in the home directory.

The aim is to provide an understanding of the system's current state and offer recommendations for optimization (if possible).

Methodologies

To gather the necessary data for this report, a Bash script was executed on the system. This script retrieved information about the network's public and private IP addresses of the users which contains, MAC address, CPU usage, memory usage, active system services, and the largest files in the home directory.

- Public and Private IP Addresses: The script used the `curl` command to fetch the public IP address and the ip command to identify the private IP address.
- MAC Address: The script used the `ip` command to find the MAC address of the network interface and masked sensitive portions of it for privacy.
- CPU Usage: The script used the `ps` command to identify the top 5 processes using the most CPU resources.
- Memory Usage: The `free` command was used to display the system's memory usage statistics, including total and available memory.

- Active System Services: The script used the `systemctl` command to list active system services and their statuses.

- Largest Files in Home Directory: The script used the `find` and `du` commands to locate the top 10 largest files in the /home directory.

Discussion

The analysis revealed several findings that are worth noting, I'll go through it in a step by step approach to better explain on each steps and flag that I've used:

```
#!/bin/bash
```

This is called the Shebang line, it tells the system what interpreter to use to run the commands written within the script.

```
separator () {
      echo "---------- ---------- ----------"
}
```

I've created a separator function for visual separation between sections for easier reading.

Now that the interpreter and easier reading formatting of the output is done, let's go in depth of the command and flag(s) used

Public and Private IP Addresses

Public IP Address

```
# The first command will be run to get and display your external/public IP
Address.
public_ip=$(curl -s ifconfig.io)
separator
      echo "Your External/Public IP Address is: $public_ip"
```

A public IP address is the outward-facing (public-facing) IP address assigned to the router by the internet service provider (ISP). The router uses its public IP to access the internet.

Other computers on the internet use your public IP address to communicate with the devices on your network

Let's break down the command `public_ip=$(curl -s ifconfig.io)`

- The fetched public IP is stored in the `public_ip` variable.
- The `$` symbol used here will place the whole command into the variable `public_ip`
    - This helps with getting the code cleaner for any user who wants to read the whole script. It's definitely easier to read a command as a variable than the command in its long form.
- The `curl` command fetches the public IP address from the ifconfig.io service.
- The `-s` flag tells curl to operate silently without outputting progress information.

```
echo "Your External/Public IP Address is: $public_ip":
```

- The `echo` command prints a message with the retrieved public IP address.
    - I used " " here as there is a variable used.
    - ' ' will forcefully output anything in the single quotes and a string.

Private IP Address

```
# The second command will be run to get and display your internal/private
IP Address.
private_ip=$(ip addr show | grep -Eo 'inet [0-9.]+' | grep -v '127.0.0.1' |
awk '{print $2}')
separator
      echo "Your Internal/Private IP Address is: $private_ip"
```

A private IP address is the address your network router assigns to your device. Each device within the same network is assigned a unique private IP address; this is how devices on the same internal network talk to each other.

Let's break down the command `private_ip=$(ip addr show | grep -Eo 'inet [0-9.]+' | grep -v '127.0.0.1' | awk '{print $2}')`

- This line uses `ip addr show` to display network interface information.
- I used the `grep` command to filter lines that contain the word `inet` followed by an IPv4 address.
    - The `-E` flag tells grep to interpret the pattern as an extended regular expression
    - The `-o` flag tells grep to only output the parts of the line that match the regular

- The `grep` command `-v '127.0.0.1'` option excludes the loopback address.
- The `awk` command extracts the second field from the filtered lines, which contains the private IP address.
- The extracted private IP is stored in the private_ip variable.

The output will look like this in the terminal.

```
---------- ---------- ----------
Your External/Public IP Address is: 86.48.10.189
---------- ---------- ----------
Your Internal/Private IP Address is: 192.168.232.128
```

MAC Address

```
# The third command will be ran to get and display your MAC Address
mac_add=$(ip addr show | grep link/ether | awk '{print $2}' | awk -F:
'{print $1 ":XX:XX:" $4 ":" $5 ":" $6}')
separator
      echo "Your MAC Address is: $mac_add"
```

The MAC address was retrieved and masked appropriately to maintain security and privacy.

Let's break down the command `mac_add=$(ip addr show | grep link/ether | awk '{print $2}' | awk -F: '{print $1 ":XX:XX:" $4 ":" $5 ":" $6}')`

- This line uses `ip addr show` to display network interface information.
- I used the `grep link/ether` command to filter lines containing the MAC address (link/ether).
- The `awk '{print $2}'` command extracts the MAC address.
- The `awk -F:` line masks the MAC address by keeping the first segment and replacing the second and third segments with XX.
- The masked MAC address is stored in the mac_add variable.

The output will look like this in the terminal

```
---------- ---------- ----------
Your MAC Address is: 00:XX:XX:49:83:1e
---------- ---------- ----------
```

CPU Usage

The top 5 processes using the most CPU resources were identified. These processes appeared to be system-related tasks.To achieve this, I used the `ps` and `aux` command with the `--sort` flag to have a better view of the top 5 processes using the most CPU resources.

Let's break down the command `ps aux --sort=%cpu | head -n 6:`

- This command uses `ps aux` to list all processes with their CPU usage (%cpu).
- The output is sorted by CPU usage in descending order using `--sort=%cpu`.
- The `head -n 6` command returns the top 6 lines from the sorted list

The output will look like this in the terminal

```
--------- --------- ---------
Below are the top 5 processes retrived from your CPU
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           3  0.0  0.0      0     0 ?        S    22:20   0:00 [pool_workqueue_release]
root           4  0.0  0.0      0     0 ?        I<   22:20   0:00 [kworker/R-rcu_g]
root           5  0.0  0.0      0     0 ?        I<   22:20   0:00 [kworker/R-rcu_p]
root           6  0.0  0.0      0     0 ?        I<   22:20   0:00 [kworker/R-slub_]
root           7  0.0  0.0      0     0 ?        I<   22:20   0:00 [kworker/R-netns]
--------- --------- ---------
```

Memory Usage

I've used the `free` command to get the memory usage of the system. The system had sufficient available memory, suggesting that it was not under significant memory pressure at the time of analysis.

Let's break down the command `free -h`

- The `free` command is used to display memory usage statistics.
- The `-h flag` formats the output in human-readable format (e.g., using units like MiB or GiB).

The output will look like this in the terminal

```
Memory Usage:
               total        used        free      shared  buff/cache   available
Mem:           1.9Gi       788Mi       770Mi       7.9Mi       566Mi       1.2Gi
Swap:          1.0Gi          0B       1.0Gi
---------- ---------- ----------
```

Active System Services

The list of active services was consistent with a standard system configuration. I used the systemctl command with multiple flags to achieve this, I'll explain below

I'll break down the command used `systemctl list-units --type=service --state=active --no-pager`

`systemctl:`

- This is the command-line interface for interacting with systemd.
    - It allows you to manage services, units, and other system components.

- `list-units:`
    - This systemctl subcommand lists system units.
    - Units are objects that systemd manages, such as services, sockets, mounts, and timers.

- `--type=service:`
    - This option filters the list to include only units of type "service".
    - Services are units that manage long-running processes.

- `--state=active:`
    - This option filters the list to include only units in the "active" state.
    - The "active" state means that the service is currently running and operational.

- `--no-pager:`
    - This option disables the use of a pager (such as less or more) to display the output.
    - Without this option, the output might be passed through a pager, which can be inconvenient for scripting or for those who prefer to see all output at once.

Here is how the output will look like in the terminal

```
---------- ---------- ----------
Active Services and Status:
  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
  ModemManager.service                     loaded active running Modem Manager
  networking.service                       loaded active exited  Raise network interfaces
  NetworkManager-wait-online.service       loaded active exited  Network Manager Wait Online
  NetworkManager.service                   loaded active running Network Manager
  open-vm-tools.service                    loaded active running Service for virtual machines hosted on VMware
  plymouth-quit-wait.service               loaded active exited  Hold until boot process finishes up
  plymouth-read-write.service              loaded active exited  Tell Plymouth To Write Out Runtime Data
  plymouth-start.service                   loaded active exited  Show Plymouth Boot Screen
  polkit.service                           loaded active running Authorization Manager
  rpc-statd-notify.service                 loaded active exited  Notify NFS peers of a restart
  rsyslog.service                          loaded active running System Logging Service
  rtkit-daemon.service                     loaded active running RealtimeKit Scheduling Policy Service
  systemd-binfmt.service                   loaded active exited  Set Up Additional Binary Formats
  systemd-journal-flush.service            loaded active exited  Flush Journal to Persistent Storage
  systemd-journald.service                 loaded active running Journal Service
  systemd-logind.service                   loaded active running User Login Management
  systemd-modules-load.service             loaded active exited  Load Kernel Modules
  systemd-random-seed.service              loaded active exited  Load/Save OS Random Seed
  systemd-remount-fs.service               loaded active exited  Remount Root and Kernel File Systems
  systemd-sysctl.service                   loaded active exited  Apply Kernel Variables
  systemd-tmpfiles-setup-dev-early.service loaded active exited  Create Static Device Nodes in /dev gracefully
  systemd-tmpfiles-setup-dev.service       loaded active exited  Create Static Device Nodes in /dev
  systemd-tmpfiles-setup.service           loaded active exited  Create Volatile Files and Directories
  systemd-udev-trigger.service             loaded active exited  Coldplug All udev Devices
  systemd-udevd.service                    loaded active running Rule-based Manager for Device Events and Files
  systemd-update-utmp.service              loaded active exited  Record System Boot/Shutdown in UTMP
  systemd-user-sessions.service            loaded active exited  Permit User Sessions
  udisks2.service                          loaded active running Disk Manager
  upower.service                           loaded active running Daemon for power management
  user-runtime-dir@1000.service            loaded active exited  User Runtime Directory /run/user/1000
  user@1000.service                        loaded active running User Manager for UID 1000

Legend: LOAD   → Reflects whether the unit definition was properly loaded.
        ACTIVE → The high-level unit activation state, i.e. generalization of SUB.
        SUB    → The low-level unit activation state, values depend on unit type.

42 loaded units listed.
---------- ---------- ----------
```

Largest Files in Home Directory

To show the largest files in the `home` directory were related to the user's activities and software installations. I used the `find`, `du`, command to extract the information out.

Lets break down the command I used `find "$HOME" -type f -exec du -h {} + | sort -rh | head -n 10`

`find "$HOME" -type f:`

- The find command searches through files and directories.

- The argument `"$HOME"` specifies the location to search, which is the user's home directory (e.g., /home/username on Linux).
- The flag `-type f` tells find to only search for regular files, ignoring directories and other types of objects.

`-exec du -h {} +:`

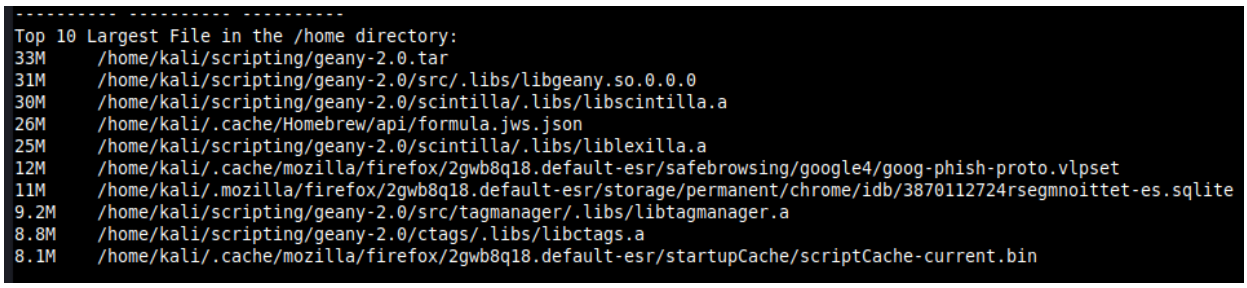
- The `-exec` flag of the find command allows you to execute a specified command on each file found.
- The `du` command provides the disk usage of the specified path (a file or directory) and its contents.
- In this case, `du -h` is the command to execute, using the `-h` flag which calculates the disk usage of each file in a human-readable format (e.g., using units like KB, MB, GB).
- The `{}` symbol is a placeholder for each file found.
- The `+` at the end allows find to pass multiple files at once to `du`, which is more efficient than calling `du` separately for each file.

`sort -rh:`

- The sort command sorts the output.
- The `-r` flag sorts the output in reverse order, from largest to smallest.
- The `-h` flag allows the sort to consider human-readable units (e.g., KB, MB, GB) when comparing sizes.

- The head command displays the first few lines of the input.
- The `-n 10` option specifies that only the first 10 lines of sorted output should be displayed.

This is how the output will look like in the terminal



Conclusion

The analysis of the computer system's network infrastructure reveals a healthy and well-configured environment. Here's a summary of the findings:

Connectivity:

The system has both a valid public and private IP address, ensuring its ability to communicate within the local network and over the internet.

Resource Utilization:

Available memory appears sufficient, and CPU usage seems to be primarily associated with expected system processes. This indicates that the system is not currently experiencing performance bottlenecks due to resource constraints.

Services:

Active system services align with a standard configuration, suggesting no unexpected or potentially problematic services in operation.

File Storage:

The largest files within the home directory appear related to the user's work or software installs. No immediate indication of space pressure exists.

Recommendations

Based on the findings, the following recommendations are provided to optimize system performance and security:

Security:

- Mask the full MAC address for privacy reasons, especially if the report will be shared or stored.

Usage:

- Please note that the code and command that was used within this script is designed for unix-based systems meaning this script will mainly work for Linux or macOS systems.
- If you are running on windows, you might have to do a linux command line installation through powershell with administrator mode turned on.

    - To install WSL: Open PowerShell as administrator and run: `wsl --install` (You might need to reboot)
    - Install a Linux distribution: Choose your preferred distro (Ubuntu, Debian, etc.) from the Microsoft Store.
    - Launch the Linux environment: Search for your installed distro in the Start menu.
    - Place your script in the Linux file system: You can access your Windows files from within the Linux environment (usually mounted under /mnt/).
    - Run the script: Navigate to the script's directory and execute it using `bash script_name.sh`

References:

- https://developer.ibm.com/articles/what-is-curl-command/
- https://www.linode.com/docs/guides/how-to-use-the-linux-ip-command/
- https://en.wikipedia.org/wiki/MAC_spoofing#:~:text=The%20process%20of%20masking%20a,computer's%20identity%2C%20for%20any%20reason.
- https://wiki.archlinux.org/title/MAC_address_spoofing
- https://www.ibm.com/docs/en/aix/7.3?topic=p-ps-command#:~:text=The%20ps%20command%20writes%20the,display%20extra%20thread-related%20columns.
- https://www.geeksforgeeks.org/ps-command-in-linux-with-examples/
- https://www.linode.com/docs/guides/use-the-ps-aux-command-in-linux/
- https://www.geeksforgeeks.org/free-command-linux-examples/
- https://gcore.com/learning/how-to-use-systemctl-command-in-linux/#:~:text=The%20systemctl%20command%20in%20Linux%20is%20a%20utility%20that%20manages,default%20on%20many%20Linux%20distributions.
- https://www.tecmint.com/list-all-running-services-under-systemd-in-linux/
- https://bbs.archlinux.org/viewtopic.php?id=276103
- https://www.redhat.com/sysadmin/du-command-options
- https://superuser.com/questions/638375/what-does-mean-in-find-in-linux
- https://man7.org/linux/man-pages/man1/sort.1.html
- https://github.com/jlevy/the-art-of-command-line