

# **Monitoring a server: How to**

## **1. Introduction**

A server monitoring report aims at giving the most accurate view of what monitoring a server managed by a Linux distro encompasses. It also wants to underline the importance of said monitoring for companies' network infrastructure to remain safe according to the CIA triad (Confidentiality, Integrity, Availability).

Since this research is done in the context of my Cybersecurity studies at Bencode, I'll be analysing the needs of a very simple network infrastructure (one server, one client) while considering that real-life needs call for greater care and attention to the intricacies a larger network requires.

## **2. Server Structure Overview**

As said above, I'll do my research on a simple point-to-point network topology as I'll be looking into a "one client – one server" architecture that is currently installed on my laptop in the form of a Debian CLI server giving DHCP, DNS and HTTP access to another Debian GUI virtual machine.

It seems interesting here to then include a brief but detailed description of the servers used. This description could tackle the below components:

- The hardware used by the server.
- The software running on it.
- The network it is included in.

## **3. Monitoring Strategy**

An efficient monitoring is dependent on a strategy designed during the uphill moments of the process. Such strategy can concentrate on multiple metrics to ensure maximum strength to the server, such as:

- CPU usage
- Memory load
- HDD/SDD space
- Network traffic
- Power usage

These are major point to be paid attention to as they are the ones whose healthiness defines the availability and reachability of a server while maintaining security. Let's dig deeper into these points:

Before digging deeper into the more specific tools, we can mention that some commands can ensure you have a first general view of the health of your system. These commands may sometimes need to be installed as they rely on non-default Linux packages. For our own projects, we will use the package "Nmon" as it shows a short and clean first look at your systems' parameters without overflowing you in info.

```
inconnu@DESKTOP-8E6S40D: x + ~
nmon-16n Hostname=DESKTOP-8E6S40 Refresh= 2secs 14:42.40

  For help type H or ...
  nmon -? - hint
  nmon -h - full details

  To stop nmon type q to Quit

Use these keys to toggle statistics on/off:
c = CPU          l = CPU Long-term      - = Faster screen updates
C = " WideView  U = Utilisation          + = Slower screen updates
m = Memory       V = Virtual memory      j = File Systems
d = Disks        n = Network              . = only busy disks/procs
r = Resource     N = NFS                  h = more options
k = Kernel       t = Top-processes       q = Quit
```

## CPU usage

MPstat (sysstat): this is a part of the “Sysstat” package that helps looking at the pressure your CPU is currently being subjected to. As you can see below, you can also give it some options so that the CPU load gets checked every x seconds you tell it to check (among other options)

```
(kali@kali)-[~]
$ mpstat
Linux 6.6.9-amd64 (kali) 04/08/2024 _x86_64_ (2 CPU)

09:06:51 AM CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
09:06:51 AM all 4.40 0.09 2.43 7.11 0.00 1.03 0.00 0.00 0.00 84.94

(kali@kali)-[~]
$ man mpstat

(kali@kali)-[~]
$ mpstat 1 5
Linux 6.6.9-amd64 (kali) 04/08/2024 _x86_64_ (2 CPU)

09:08:03 AM CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
09:08:04 AM all 1.82 0.00 0.91 0.00 0.00 0.00 0.00 0.00 0.00 97.27
09:08:05 AM all 2.94 0.00 0.49 0.00 0.00 0.49 0.00 0.00 0.00 96.08
09:08:06 AM all 1.98 0.00 1.49 0.00 0.00 0.50 0.00 0.00 0.00 96.04
09:08:07 AM all 2.03 0.00 3.05 0.51 0.00 0.51 0.00 0.00 0.00 93.91
09:08:08 AM all 1.52 0.00 2.02 1.01 0.00 0.00 0.00 0.00 0.00 95.45
Average: all 2.06 0.00 1.57 0.29 0.00 0.29 0.00 0.00 0.00 95.79
```

## Memory load

The “free” command can help you see the overall memory details of your current distro, although not hugely useful as you can’t pinpoint any apps that would/could take a huge portion of your available memory:

```
(kali@kali)-[~]
$ free -h
              total        used        free      shared  buff/cache   available
Mem:          1.9Gi         850Mi         541Mi         20Mi         760Mi         1.1Gi
Swap:          1.0Gi           0B          1.0Gi
```

What can help you get more detailed info though is the “top” command, which can then list the entire list of processes being active on your machine while telling you the amount of memory used by them and which user triggered them:

```
top - 18:09:12 up 1:16, 1 user, load average: 0.27, 0.32, 0.25
Tasks: 188 total, 3 running, 132 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.8 us, 2.2 sy, 0.0 ni, 95.7 id, 0.0 wa, 0.0 hi, 0.3 si, 0.0 st
MiB Mem : 1976.8 total, 537.5 free, 853.9 used, 766.9 buff/cache
MiB Swap: 1928.8 total, 1928.8 free, 0.0 used, 1122.8 avail Mem

  PID USER      NI  VIRT  RES  SHR S  CPU% MEM%   TIME+  COMMAND
 693 root        0 515976 173880 68076 R   1.8  8.6  1:22.34 /usr/lib/xorg/Xorg :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch
1801 kali        0 466576 188280 91452 S   1.1  5.3  0:51.22 /usr/bin/terminal
1979 kali        0 385264 52492 22036 S   0.6  2.6  0:27.27 /usr/lib/x86_64-linux-gnu/xfce/panel/wrapper-2.0 /usr/lib/x86_64-linux-gnu/xfce/panel/plugins/libcpugraph.so 13 27262988 cpugraph CPU Graph Graphical representat
1881 kali        0 342216 38080 21312 S   0.5  1.5  0:18.63 /usr/lib/x86_64-linux-gnu/xfce/panel/wrapper-2.0 /usr/lib/x86_64-linux-gnu/xfce/panel/plugins/libgemon.so 15 27262998 gemon Generic Monitor Show output of a co
1823 kali        0 980780 218824 79436 S   0.4  3.3  0:21.19 xfwm --display :0.0 --sm-client-id 2321eeeb-dbf4-4d93-b57c-b2f1d97d623
1 root          0 22280 12752 9552 S   0.1  0.6  0:01.64 /sbin/init splash
958 kali        0 217984 2944 2688 S   0.1  0.1  0:10.80 /usr/bin/XdixClient --draganddrop
985 kali        0 217576 3456 2944 S   0.1  0.2  0:01.21 /usr/bin/XdixClient --wmgx
37569 kali       0 21888 5760 3584 R   0.1  0.3  0:00.16 top
17 root         0 0 0 0 R   0.1  0.0  0:02.22 [rcu_preempt]
518 root        0 8284 7512 1664 S   0.1  0.4  0:00.56 /usr/sbin/havedd --foreground --verbose=1
616 root        0 293816 3408 3872 S   0.1  0.2  0:01.33 /usr/sbin/XdixService
967 kali        0 217468 3872 2688 S   0.1  0.2  0:00.95 /usr/bin/XdixClient --seamless
1888 kali        0 385728 38080 20848 S   0.1  1.5  0:01.78 xfsettingd --display :0.0 --sm-client-id 205069710-83bf-420e-9551-39084280135e
1123 kali        0 455812 41912 33480 S   0.1  2.1  0:01.19 /usr/lib/x86_64-linux-gnu/xfce/notifyd/xfce-notifyd
11880 root        0 0 0 0 I   0.1  0.0  0:00.97 [kworker/0:0-events]
25128 root        0 0 0 0 I   0.1  0.0  0:00.00 [kworker/1:2-events]
2 root          0 0 0 0 S   0.5  0.0  0:00.00 [kthread]
3 root          0 0 0 0 S   0.5  0.0  0:00.00 [pool_workqueue_release]
4 root          0 -20 0 0 I   0.1  0.0  0:00.00 [kworker/R-rcu_p]
5 root          0 -20 0 0 I   0.1  0.0  0:00.00 [kworker/R-rcu_p]
6 root          0 -20 0 0 I   0.1  0.0  0:00.00 [kworker/R-rcu_p]
7 root          0 -20 0 0 I   0.1  0.0  0:00.00 [kworker/R-netns]
18 root         0 -20 0 0 I   0.1  0.0  0:00.00 [kworker/0:0-events_highpri]
11 root         0 0 0 0 I   0.1  0.0  0:00.00 [kworker/R-netns-rx-conversion]
12 root         0 -20 0 0 I   0.1  0.0  0:00.00 [kworker/R-mm_pg]
13 root         0 0 0 0 I   0.1  0.0  0:00.00 [rcu_tasks_kthread]
14 root         0 0 0 0 I   0.1  0.0  0:00.00 [rcu_tasks_rude_kthread]
15 root         0 0 0 0 I   0.1  0.0  0:00.00 [rcu_tasks_trace_kthread]
16 root         0 0 0 0 S   0.5  0.0  0:00.00 [ksftirq/0]
18 root         0 0 0 0 S   0.5  0.0  0:00.00 [migration/0]
19 root        -51 0 0 0 S   0.5  0.0  0:00.00 [idle_inject/0]
20 root         0 0 0 0 S   0.5  0.0  0:00.00 [cpuhp/0]
21 root         0 0 0 0 S   0.5  0.0  0:00.00 [cpuhp/1]
22 root         0 0 0 0 S   0.5  0.0  0:00.00 [idle_inject/1]
23 root         0 0 0 0 S   0.5  0.0  0:00.00 [migration/1]
24 root         0 0 0 0 S   0.5  0.0  0:00.00 [ksftirq/1]
26 root         0 0 0 0 I   0.1  0.0  0:00.00 [kworker/u5:0-events_unbound]
27 root         0 0 0 0 I   0.1  0.0  0:00.00 [kworker/u5:0-events_unbound]
30 root         0 0 0 0 I   0.1  0.0  0:00.00 [kworker/u6:1-events_unbound]
31 root         0 0 0 0 S   0.5  0.0  0:00.00 [kdestmap]
32 root         0 -20 0 0 I   0.1  0.0  0:00.00 [kworker/R-net]
33 root         0 0 0 0 I   0.1  0.0  0:00.00 [kuidfd]
34 root         0 0 0 0 I   0.1  0.0  0:01.06 [kworker/u5:1-flush-810]
35 root         0 0 0 0 S   0.5  0.0  0:00.00 [khungtaskd]
36 root         0 0 0 0 S   0.5  0.0  0:00.00 [oom_reaper]
37 root         0 -20 0 0 I   0.1  0.0  0:00.00 [kworker/R-write]
38 root         0 0 0 0 S   0.5  0.0  0:00.22 [krmockd0]
39 root         0 0 0 0 S   0.5  0.0  0:00.00 [hissd]
```

A shorter output can be caught with the “ps” command, as it’ll only give you a snapshot of the running processes without giving you all the “hidden” processes running :

```
(kali㉿kali)-[~]
$ ps
  PID TTY          TIME CMD
 1804 pts/0        00:00:17 zsh
 53188 pts/0        00:00:00 ps
```

You can then use the “lsOf” command and the Process ID (PID) number to target a specific program and see all its dependencies’ memory use:

```
(kali㉿kali)-[~]
$ lsOf -p 1801
COMMAND  PID USER  FD   TYPE    DEVICE  SIZE/OFF      NODE NAME
qterminal 1801 kali   cwd   DIR      8,1      4096    917506 /home/kali
qterminal 1801 kali   rtd   DIR      8,1      4096         2 /
qterminal 1801 kali   txt   REG      8,1    485608 4353522 /usr/bin/qterminal
qterminal 1801 kali   DEL   REG      0,1         116 /memfd:xorg
qterminal 1801 kali   mem   REG      8,1   4033420 3685399 /usr/share/fonts/truetype/droid/DroidSansFallbackFull.ttf
qterminal 1801 kali   mem   REG      8,1   24061128 3545859 /usr/lib/x86_64-linux-gnu/libz3.so.4
qterminal 1801 kali   mem   REG      8,1  124536680 3546295 /usr/lib/x86_64-linux-gnu/libLLVM-17.so.1
qterminal 1801 kali   mem   REG      8,1   708920 3685456 /usr/share/fonts/truetype/dejavu/DejaVuSans-Bold.ttf
qterminal 1801 kali   DEL   REG      0,1         117 /memfd:xorg
qterminal 1801 kali   mem   REG      8,1   32948200 3545562 /usr/lib/x86_64-linux-gnu/dri/swrast_dri.so
qterminal 1801 kali   DEL   REG      0,1         1109 /memfd:xorg
qterminal 1801 kali   mem   REG      8,1   320028 3685360 /usr/share/fonts/truetype/firacode/FiraCode-Bold.ttf
qterminal 1801 kali   mem   REG      8,1    6160 3546252 /usr/lib/x86_64-linux-gnu/libxshmfence.so.1.0.0
qterminal 1801 kali   DEL   REG      0,1         1108 /memfd:xorg
qterminal 1801 kali   mem   REG      8,1   1770584 3544979 /usr/lib/x86_64-linux-gnu/libxml2.so.2.9.14
qterminal 1801 kali   mem   REG      8,1   22816 3545647 /usr/lib/x86_64-linux-gnu/libXxf86vm.so.1.0.0
qterminal 1801 kali   DEL   REG      0,1         1107 /memfd:xorg
qterminal 1801 kali   mem   REG      8,1   510612 3685435 /usr/share/fonts/truetype/noto/NotoSansMono-Regular.ttf
qterminal 1801 kali   mem   REG      8,1   147248 3544949 /usr/lib/x86_64-linux-gnu/libdrm_intel.so.1.0.0
qterminal 1801 kali   mem   REG      8,1   216368 3544643 /usr/lib/x86_64-linux-gnu/libtinfo.so.6.4
qterminal 1801 kali   mem   REG      8,1   216656 3544623 /usr/lib/x86_64-linux-gnu/libedit.so.0.2.0.72
qterminal 1801 kali   mem   REG      8,1   49180 4337902 /usr/share/icons/hicolor/icon-theme.cache
qterminal 1801 kali   mem   REG      8,1   43472 3545039 /usr/lib/x86_64-linux-gnu/libpciaccess.so.0.11.1
qterminal 1801 kali   mem   REG      8,1   39216 3544665 /usr/lib/x86_64-linux-gnu/libdrm_nouveau.so.2.0.0
```

## HDD/SDD disk usage

After install the “iotop” package, you’ll then be able to have an overview of your disks’ write & read activities in a very precise way:

Total DISK READ:		0.00 B/s	Total DISK WRITE:		0.00 B/s
Current DISK READ:		0.00 B/s	Current DISK WRITE:		0.00 B/s
TID	PRI	USER	DISK READ	DISK WRITE	COMMAND
1	be/4	root	0.00 B/s	0.00 B/s	init splash
2	be/4	root	0.00 B/s	0.00 B/s	[kthreadd]
3	be/4	root	0.00 B/s	0.00 B/s	[pool_workqueue_release]
4	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-rcu_g]
5	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-rcu_p]
6	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-slub_]
7	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-netns]
10	be/0	root	0.00 B/s	0.00 B/s	[kworker/0:0H-events_highpri]
11	be/4	root	0.00 B/s	0.00 B/s	[kworker/u4:0-ext4-rsv-conversion]
12	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-mm_pe]
13	be/4	root	0.00 B/s	0.00 B/s	[rcu_tasks_kthread]
14	be/4	root	0.00 B/s	0.00 B/s	[rcu_tasks_rude_kthread]
15	be/4	root	0.00 B/s	0.00 B/s	[rcu_tasks_trace_kthread]
16	be/4	root	0.00 B/s	0.00 B/s	[ksoftirqd/0]
17	be/4	root	0.00 B/s	0.00 B/s	[rcu_preempt]
18	rt/4	root	0.00 B/s	0.00 B/s	[migration/0]
19	rt/4	root	0.00 B/s	0.00 B/s	[idle_inject/0]
20	be/4	root	0.00 B/s	0.00 B/s	[cpuhp/0]
21	be/4	root	0.00 B/s	0.00 B/s	[cpuhp/1]
22	rt/4	root	0.00 B/s	0.00 B/s	[idle_inject/1]
23	rt/4	root	0.00 B/s	0.00 B/s	[migration/1]
24	be/4	root	0.00 B/s	0.00 B/s	[ksoftirqd/1]
27	be/4	root	0.00 B/s	0.00 B/s	[kworker/u5:0-events_unbound]
30	be/4	root	0.00 B/s	0.00 B/s	[kworker/u6:1-events_unbound]
31	be/4	root	0.00 B/s	0.00 B/s	[kdevtmpfs]
32	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-inet_]
33	be/4	root	0.00 B/s	0.00 B/s	[kauditd]
34	be/4	root	0.00 B/s	0.00 B/s	[kworker/u5:1-events_unbound]
35	be/4	root	0.00 B/s	0.00 B/s	[khungtaskd]
36	be/4	root	0.00 B/s	0.00 B/s	[oom_reaper]
37	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-write]
38	be/4	root	0.00 B/s	0.00 B/s	[kcompactd0]
39	be/5	root	0.00 B/s	0.00 B/s	[ksmd]
40	be/7	root	0.00 B/s	0.00 B/s	[khugepaged]
41	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-kinte]
42	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-kbloc]
43	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-blkc]
44	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-tpm_d]
45	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-dac-]
46	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-devfr]
48	be/0	root	0.00 B/s	0.00 B/s	[kworker/0:1H-kblockd]
49	be/4	root	0.00 B/s	0.00 B/s	[kswapd0]
57	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-kthro]
59	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-acpi_]
60	be/0	root	0.00 B/s	0.00 B/s	[kworker/R-mld_]

Should you want less information to get a quick overview, the “iostat” command can then help you for this:

(kali@kali)-[~]						
\$ iostat -h						
Linux 6.6.9-amd64 (kali)		04/08/2024		_x86_64_	(2 CPU)	
avg-cpu:	%user	%nice	%system	%iowait	%steal	%idle
	2.5%	0.0%	2.3%	1.3%	0.0%	93.8%
tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd Device
5.05	126.0k	86.7k	0.0k	678.4M	467.2M	0.0k sda

## Network traffic

The “iftop” command, after being installed, can help you display the current in- and outgoing data flow of the operating system currently running.

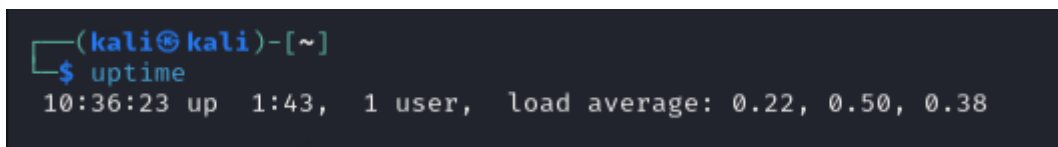


However, if you'd like a finer tool to work with, the “ss” command can help you see which process of your computer tried to use which port:



## Power usage

On this matter, it may be interesting to also know as of when your machine was running (since when there's an uptime). For this, the very logical "uptime" command will help you:



After getting this info, you can retrieve the proper power usage information with the “Powertop” package :

File Actions Edit View Help			
PowerTOP 2.15 Overview Idle stats Frequency stats Device stats Tunables WakeUp			
Summary: 183.2 wakeups/second, 0.0 GPU ops/seconds, 0.0 VFS ops/sec and 9.9% CPU use			
Usage	Events/s	Category	Description
2.0 ms/s	66.8	Timer	tick_sched_timer
14.3 ms/s	16.6	Process	[PID 692] /usr/lib/xorg/Xorg :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch
2.1 ms/s	17.4	Process	[PID 965] /usr/bin/VBoxClient --draganddrop
385.5 µs/s	15.9	Process	[PID 17] [rcu_preempt]
4.2 ms/s	7.5	Process	[PID 1023] xfwm4 --display :0.0 --sm-client-id 23261ee08-dbf4-4d93-b57c-b2f10a97d623
1.1 ms/s	9.9	Timer	hrtimer_wakeup
54.3 µs/s	6.9	KWork	psi_avg5_work
5.6 ms/s	2.0	Process	[PID 1081] /usr/lib/x86_64-linux-gnu/xfce4/panel/wrapper-2.0 /usr/lib/x86_64-linux-gnu/xfce4/panel/plugins/libgenmon.so 15 2726
7.1 ms/s	2.6	Process	[PID 1879] /usr/lib/x86_64-linux-gnu/xfce4/panel/wrapper-2.0 /usr/lib/x86_64-linux-gnu/xfce4/panel/plugins/libcpugraph.so 13 27
6.7 ms/s	0.00	Timer	vbg_heartbeat_timer
1.0 ms/s	1.5	Process	[PID 954] /usr/bin/VBoxClient --seamless
1.7 ms/s	2.5	Process	[PID 1881] /usr/bin/gterminal
2.4 ms/s	1.6	Process	[PID 47984] /usr/lib/firefox-esr/firefox-esr
268.5 µs/s	2.1	Process	[PID 952] /usr/bin/VBoxClient --seamless
430.1 µs/s	1.7	Process	[PID 48542] /usr/lib/firefox-esr/firefox-esr -contentproc -childID 3 -isForBrowser -prefsLen 26695 -prefMapSize 233950 -jsInitl
178.8 µs/s	1.8	Process	[PID 68009] powertop
37.0 µs/s	1.8	KWork	delayed_vfree_work
2.0 ms/s	0.7	Interrupt	[7] sched(softirq)
41.7 µs/s	1.3	Process	[PID 16] [ksoftirqd/0]
131.1 µs/s	1.2	Process	[PID 987] /usr/bin/VBoxClient --vmsvga
180.0 µs/s	1.1	Process	[PID 988] /usr/bin/VBoxClient --vmsvga
463.7 µs/s	0.9	Interrupt	[20] vboxguest
33.4 µs/s	1.0	Process	[PID 38] [kcompactd0]
1.0 ms/s	0.6	Process	[PID 48496] /usr/lib/firefox-esr/firefox-esr -contentproc -childID 3 -isForBrowser -prefsLen 26695 -prefMapSize 233950 -jsInitl
687.5 µs/s	0.6	Process	[PID 1084] /usr/lib/x86_64-linux-gnu/xfce4/panel/wrapper-2.0 /usr/lib/x86_64-linux-gnu/xfce4/panel/plugins/libxfcepowermanager
0.9 ms/s	0.5	Process	[PID 1052] /usr/libexec/upowerd
16.9 µs/s	0.8	KWork	vmstat_shepherd
54.1 µs/s	0.6	Process	[PID 1628] /usr/libexec/gvfs-afc-volume-monitor
1.2 ms/s	0.15	Process	[PID 5871] /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only

Although it seems to contain some errors as I can't yet retrieve the mw/h data for each process, but rather a ms/s value.

## Users' activities

Lastly, we need to know where to look for the logs, what happened during our server's session, who did what, etc.

All logs being in the "var/log" folder, we can dig into them to see the multiple apps' activity history.

As for the users, we can check multiple data concerning them :

- When they were connected thanks to the "W" or "Finger" command (to be installed first)
- When the last sessions were running with the "last" command
- With the output of the "W" or "Finger" command, we can have a deeper look at the commands that were run by certain users after typing "PS -ef | grep ^username"

## 4. Monitoring logs and alerts

I'll mention quickly here two aspects of the logs and alerts that are important to consider when monitoring a server: their setup and their frequencies.

- Setup: one needs to ensure that the alerts are correctly configured as to ensure that the incident team won't need to filter through countless alerts, thus increasing the possibility of missing an important one.
- Frequency: logs and alerts need to be monitored live, but a manual human check may be done fewer times (daily)

## 5. Conclusion

Although the above theoretical documentation doesn't encompass all the available tools that can perform monitoring activities, and maybe not always the most efficient one, it gave a first look at the commands and programs that may help one monitor efficiently and speedily the activities of a server.