Linux Incident Response: Finding Malware with /Proc

Taz Wake | Halkyn Consulting Ltd | Originally published 2026-07-02

It is one of the most common tricks we see in Linux intrusions: the attacker drops a payload, executes it, then immediately deletes the file from disk. A disk-based scan finds nothing. A file integrity check finds nothing. The attacker is betting that if the binary isn't on the filesystem, nobody will find it. The fun thing is that this is actually surprisingly effective.

But it doesn't have to work for the attackers, we just need to look closer.

On Linux-based operating systems, deleting a running file doesn't actually remove it. The directory entry is gone, yes, but the inode and data blocks remain allocated for as long as the process holds them open. We can find the complete binary via the link in /proc.

What does proc give us?

Every running process has a folder under /proc named for its PID, and it is one of the richest live-response evidence sources on any system. Start with the executable link:

If it looks like this:

```
root@reponse:~# ls /proc/1306/exe -lh
lrwxr-xr-x 1 root root 0 Jul  2 22:50 /proc/1306/exe -> /dev/shm/kworker (deleted)
```

Then you have a running process where the file backing the process has been deleted. Using the long format means you get some other useful information as well, such as the process start time and its owner, as well as the path to the binary on disk.

When you check processes, the things to check include running from weird locations (/dev/shm, cache folders, and rarely /tmp), or weird process names. However, the most obvious clue that something is unusual is the string "(deleted)"... It is definitely uncommon for a legitimate process to be backed by a deleted file.

The bad news is that, in this state, you won't find the file when you use OS tools to look in the folder. In the example above, ls -al /dev/shm will not return the "kworker" file. The filesystem will treat it as deleted. You *might* see an **mtime** or **ctime** change on the parent folder, but that is basically it.

The good news is that you can recover this file from disk pretty easily. We just need to make sure we follow the links in /proc. This means there are a LOT of things we can do.

Examining the file on disk

There are lots of things, but we can narrow this down to some of the more useful things. Just please note, this is not an exhaustive list, and imaginative use of the command line can work wonders.

File statistics

This gives you a way to get the stat command output even though the file isn't showing up on disk. You will get output that reflects the fact that the file is deleted (such as 0 links).

```
root@reponse:~# stat -L /proc/1306/exe
  File: /proc/1306/exe
  Size: 58232           Blocks: 120        IO Block: 4096   regular file
Device: 802h/2050d      Inode: 918274      Links: 0
Access: (0755/-rwxr-xr-x)  Uid: ( 1000/    user)   Gid: ( 1000/    user)
Access: 2026-07-02 22:50:12.000000000 +0100
Modify: 2026-07-02 22:49:58.000000000 +0100
Change: 2026-07-02 22:50:01.000000000 +0100
 Birth: 2026-07-02 22:49:58.000000000 +0100
```

Recovering the deleted file

You cant (normally) just cat the now-deleted file from disk, but that link still helps. If you want to recover the file for analysis, you have a couple of choices. My preference is getting cat out:

An alternative approach, which lots of responders I've worked with prefer, is to use dd.

You can also use cp for this:

For some responders, this is the preferred approach. The moral of this story is pick what works for you.

Where it won't work

This is not foolproof. While it works for binaries that are executed then deleted, there are (as always) exceptions around how things work.

The most common reason this approach fails is when the binary is loaded via memfd or similar. This can include using:

- memfd_create
- LD_PRELOAD from a memfd
- execve ("/proc/self/fd/X") or an anonymous file descriptor

This results in the file looking like:

And this means there is no binary on disk at all, so nothing for our tools to recover. If this has happened, you need to think about extracting all the memory-mapped sections for the binary. Something like this would be a better approach:

Another issue arises if the attacker overwrites the binary on disk before deleting it. This can cause major issues for extraction!

Script this

Like all good things in Linux, we need to think about scripting it wherever possible.

Here is an example of a bash script you can use to recover executables (or trigger via SOAR):

You can script the maps extraction with something like this:

Ultimately, you can build this into a more one-script-to-rule-them-all approach, and you can find an example of this on GitHub at: https://for577.com/proc_rec.

Hunting deleted files

The last bit of code I want to talk about is how you can use this to find anomalies in a Linux environment. Remember, deleted files backing running processes is kind of strange. It isn't guaranteed malicious, but it is worth a check, and it should be rare enough that you won't drown in false positives.

A simple approach for this is to run:

Where to go from here

This is a short(ish) post, so it can't ever cover every possible scenario. Remember, one of the best things about Linux is how we can do things in countless ways. Adversaries know we can hunt for them using these techniques, so we occasionally see other things like namespaces being used to hide the exe paths, extensive tempfs use and so on. This is all material for a future post! The good news is it is pretty rare compared to the basics we've covered here.

Overall, this is one small corner of Linux incident response. If you want to know more about this and other Linux DFIR topics, then check out https://sans.org/for577. SANS FOR577: Linux Incident Response and Threat Hunting spends six days building this capability systematically — from command-line fundamentals and attack techniques, through filesystem forensics, logging, memory collection, and enterprise-scale response,

ending with a team capstone investigating a realistic APT intrusion across a multi-system Linux environment.

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