CVE-2025-6018 & CVE-2025-6019 for Incident Responders - with hunting strings...

Taz Wake | Halkyn Consulting Ltd | Originally published 2025-06-25

A frustratingly common discussion in cybersecurity is how "secure" Linux is. I hear this all the time. Linux is great, I use it a lot, I teach a class about it, but, and this is important, there's **no such thing as a "secure OS**." It is a meaningless term. Attackers are constantly finding new ways in, and frankly, they access Linux systems *all the time*. When they do, our job is to find them, kick them out, and stop it from happening again.

Recently, the Qualys Threat Research Unit (TRU) uncovered **two critical Linux privilege escalation (LPE) flaws, CVE-2025-6018 and CVE-2025-6019**, that, when chained together, allow an **unprivileged attacker to achieve full root access** on a major Linux distribution (SUSE). On non-SUSE systems, the risk is different, but the udisks LPE is still pretty significant and affects pretty much every Linux distro.

Hopefully you have now patched, so its probably time to look at investigating to see if they were exploited.

Summary of CVE-2025-6018 and CVE-2025-6019

These two CVEs are particularly nasty because they can create a straightforward path from an unprivileged local user to full root.

- **CVE-2025-6018: LPE from unprivileged to "allow_active" in SUSE 15's PAM**
- **CVE-2025-6019: LPE from "allow_active" to root in libblockdev via udisks**

For SUSE (OpenSUSE/SUSE Enterprise Linux), the critical takeaway here is the **chaining**: CVE-2025-6018 provides the initial "allow_active" access, which then directly enables CVE-2025-6019 to go straight to root. This means a purely unprivileged local attacker can gain full root access with minimal effort.

On non-SUSE platforms, there is less direct impact because it isn't a zero-root chain. In general, "allow_active" is assigned to any user who has an *active local session* (i.e. someone logged in at the machine’s physical console, on a TTY or local X/Wayland seat). Remote or background (inactive) sessions don’t qualify. So if the user SSHs in, they won't get it, but if they use RDP/VNC-type connections, there is a good chance they'll get this.

Other techniques that can work include stealing the session of a logged-in user, sneaking into an autologin or kiosk-type session, stealing credentials and connecting via RDP (etc), or even tricking the system by spawning a fake local session with something like:

What is the impact?

When an attacker achieves root access, it is basically game over for the system. They can perform actions like:

- **Silently unloading Endpoint Detection and Response (EDR) agents**. This is a huge deal, as it blinds your security tools to their subsequent actions. This is pretty effective in Linux, where most EDR/Endpoint Protection tools run outside the kernel anyway.
- **Implanting kernel-level backdoors for persistent code execution**. This means they can maintain access even after reboots or attempts to clean the system.
- **Rewriting system configurations** that survive reboots.
- **Masking their activities**: Rootkits are designed for stealth, manipulating core OS components to hide their presence, files, processes, and network connections. This makes detection incredibly difficult.

Although not always Linux-specific, we have lots of real-world examples of rootkit usage that highlight these dangers:

- **Lazarus Group's FudModule rootkit** was used to suspend protected processes of security software like Microsoft Defender and CrowdStrike Falcon, enabling stealthy operations to target cryptocurrency professionals.
- **CosmicStrand UEFI rootkit**: This persistent firmware rootkit embedded itself in motherboard firmware, hijacking the Windows boot process *before* the OS or security tools loaded. It allowed attackers to implant additional payloads post-boot, gaining full access without even writing to disk. This highlights that simply reinstalling the OS might not be enough if the firmware is compromised.
- **Kinsing malware**: This group exploited vulnerabilities to deploy rootkits alongside cryptocurrency miners, disabling security features and deleting logs to maintain persistence and evade detection.
- **Necurs botnet**: This botnet operated for years using a kernel-mode rootkit to intercept file/registry queries, block AV updates, and conceal its services, persisting through reboots and re-infecting systems.

These new CVEs create similar opportunities. Any system with an initial local access point (like SSH) could become a launchpad for lateral movements across your network.

Investigating linux...

Threat Hunting Tips and Ideas

Linux incident response can be challenging because there's often **less evidence, it can be harder to find, and harder to interpret** compared to Windows environments. If an attacker is able to chain these exploits and get a rootkit into the system, it is going to be even harder.

With an investigation, some areas to start with are

1. SSH Authentication Logs

Location: /var/log/auth.log or /var/log/secure

Look for:

- Unusual SSH connections from unexpected IP addresses or user accounts.
- Sessions where pam_systemd incorrectly assigned XDG_SEAT or XDG_VTNR variables to remote users.
- Modifications to ~/.pam_environment files (used to spoof physical session variables).

2. D-Bus/UDisks2 Activity Logs

Location: journal entries (journalctl)

Evidence Focus:

- Unauthorized udisks2 method calls (e.g., ModifyDevice) from non-privileged users.

- Suspicious libblockdev operations triggered via D-Bus interface.
- allow_active privilege abuse in polkit rules (check /etc/polkit-1/rules.d/)

Example Journal Searches (*Note, there can be a high false positive rate with these searches. It is important that someone who knows what they are looking for carries out the searching. Also the exact syntax might need tweaking to work on yoursystem.*)

- Search for all udisks2-related entries

- Search for D-Bus udisks2 interface calls

- Combined search with case-insensitive matching

- Search with priority filtering (errors and warnings)

- Real-time monitoring for suspicious activity

- Search a specific time range with detailed output

Example Polkit Checks

- Check polkit rules files for udisks2 configurations

- Examine specific allow_active rules

3. Process Execution Traces

Sources: Auditd logs (/var/log/audit/audit.log) or Sysmon for Linux (Syslog or Journal). This can be very hard to use if you haven't prepared properly.

Evidence Focus:

- udisksd daemon spawning unexpected child processes (e.g., shells or payloads).
- Privilege escalation patterns: sudo/su attempts following udisks2 activity.
- Execution of libblockdev-related binaries with root permissions

It is harder to provide example query strings here, as it absolutely depends on how auditing or sysmon have been configured.

If you have set up Auditd in keeping with the guidance from FOR577 (https://sans.org/for577) or Rohit N's Medium post (https://izyknows.medium.com/linux-auditd-for-threat-detection-d06c8b941505), then you can search for tags, which makes it a lot easier.

> *The default audit.rules file for every Linux distro is not going to be very effective for this, so please note that the example searches here assume you have a good audit.rules file.*

This list is not-exhaustive, and is really just here to give some ideas and starting points for your threat hunting/IR activities.

- Search for udisksd process execution and child processes

- Look for udisksd spawning shells or suspicious binaries

- More targeted search for udisksd child processes with some commonly abused strings

- Search for process tree anomalies involving udisksd, relying on lazy attackers

- Search for sudo/su attempts following udisks2 activity (within time window)

- Look for privilege escalation patterns

- Search for setuid execution after udisks2 calls

- Combined search for escalation timeline

More complex searches

- Search for libblockdev-related executions with root privileges

- Look for specific libblockdev utilities

- Search for library loading and execution

- Network activity correlation with udisks2 processes

Bringing out the big guns

- Timeline correlation: udisks2 -> privilege escalation (not LinkedIn breaks the formatting a bit here)

One search to rule them all

> *Always remember to be wary of false positives with all this. The reason attackers like this type of exploit is that it can be hard to tell signal from noise.*

4. File System Artifacts

Paths to Investigate:

- /etc/pam.d/ (misconfigured PAM modules).
- /usr/libexec/udisks2/ (modified udisks2 executables or scripts).
- /tmp or /dev/shm (temporary exploit payloads) (less often /var/tmp)

As anyone who has seen my recent talks or attended a class with me will know, /dev/shm is **frequently** abused by threat actors. The /var/tmp folder is less often used, possibly because this is normally a "real" part of the filesystem, backed by physical storage, and can be captured in disk images.

What to look for:

- Timestamp anomalies in PAM or udisks2 configuration files.
- Unexpected files in home directories (e.g., .pam_environment exploitation scripts).
- Suspicious or unexpected files in temporary directories.
- Often, anything, in /dev/shm...

Notes on audit.rules

Because preparation is the most important part of an IR cycle, it would make sense to ensure that your audit.rules files are configured as expected. While building a set from scratch is outside the scope of this, you can use the Medium post mentioned above or Florian Roth's configuration to get into a good place.

At the bare minimum for this article, you need to have rules similar to:

- -a always,exit -S execve -k execution (process execution monitoring)
- -w /bin/su -p x -k privilege_escalation
- -w /usr/bin/sudo -p x -k privilege_escalation
- -a always,exit -F arch=b64 -S setuid,setgid,setreuid,setregid -k privilege_escalation

Florian Roth's audit.rules are at: https://github.com/Neo23x0/auditd

If you want to know more about investigating Linux devices and Linux IR in general, have a look at the **FOR577 - Linux Incident Response and Threat Hunting** class from SANS: 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