SYSCTL

Sysctl position paper regarding multiple Linux vulnerabilities with high impact

Version 1.0.1
2026-05-15
info@sysctl.se

Table of Contents

Revision history

Version	Date	Who	What
1.0.1	2026-05-14	rom@sysctl.se	Minor adjustments, added TL;DR
1.0	2026-05-08	rom@sysctl.se	Initial release

1 TL;DR

Many new vulnerabilities are currently being uncovered, not seldom found with help of new AI based tools. Often the vulnerabilities are bugs that have been in the code for many years, even decades, before they was found. The severity of the bugs are often high, but not critical since they are themselves not remotely exploitable. Thus many stakeholders don't think its word the time and effort to handle some of these bugs. This is the wrong conclusions since many attacks are built using chained exploits, thus something that allow remote access can be combined with a bug that gives local privilege escalation.

This document describe how Sysctl have been working to proactively mitigate these types of bugs and bug clases by hardening OS, subsystems and applications.

2 Summary

The Linux operating system, kernel and associated application ecosystem, are a foundational part of products – like Impex and deaddrop – delivered by Sysctl as virtual appliances or embedded systems. The current rate at which new vulnerabilities are discovered in low-level kernel subsystems is alarming, and the rate seem to be growing. These vulnerabilities demonstrate that modern defensive strategies can no longer rely solely on patching or vulnerability elimination, there also need to be some proactive preparation and protection, often called often called hardening.

The hardening implemented by security specialists at Sysctl is to handle situations like this by providing additional protection barriers, if a component, including the kernel, might have severe vulnerabilities. Systems must be designed under the assumption that severe vulnerabilities will exist in foundational software components. The security value of hardening is not that vulnerabilities disappear, but that exploitability, reachability, privilege escalation and blast radius are significantly reduced.

The purpose of modern hardening is to create and enforce multiple independent security boundaries within the system. This document describes the current situation, the general problem, and what Sysctl do to mitigate the problems.

3 Background

The recent publication of *page cache* vulnerabilities and exploits have caused concern among defenders, mostly due to the fact that these have been published before any patches have been released. The vulnerabilities in question are the following:

- **Copyfail** (CVE-2026-31431, first publicly discussed 29th of April 2026)
- **Dirtyfrag** (CVE-2026-43284, CVE-2026-43500, first publicly discussed 8th May 2026)

These vulnerabilities have been exploited to provide *local privilege escalation*, LPE[1]. The proof-of-concept attack code (POC) and demonstrations of these exploits[1, 2] give the perception that there is little that defenders and vendors can do to prevent them. This is, however, an oversimplification. There are mitigations and protections available as proactive security controls. And there are patches and security upgrades as a reactive measure.

They all use one of the many `address_families(7)` [1] to interact with the kernel. The *Copyfail* vulnerability utilizes the AF_ALG *socket*[2] family to bind a socket of type AF_ALG with the `aead autencesn` arguments. This is a prerequisite for the exploit to work and can be performed by unprivileged users in a Linux system. The socket call supports many legacy communication technologies[2], with legacy code.

Dirty frag [3] uses similar techniques, either by creating a *netlink socket* or a *RXRPC socket*. In some cases, the vulnerable code requires that the user running the exploit has additional capabilities, which is solved by creating a new namespace where these capabilities exist.

There have recently been additional, similar bugs, like the:

- **CVE-2026-31532** [11] a use-after-free in Controller Area Network (CAN) raw sockets vulnerability, and
- **CVE-2026-31694** [12] a page cache overflow in the Filesystem in Userspace (FUSE) vulnerability,
- **Copy Fail 2**[4], which is a bug of the same class as Copy Fail (CVE-2026-31431) but in a different subsystem.
- **Fragnesia**[20], another *universal Linux local privilege escalation exploit* which is a bug in the Dirty frag bug class

Once a bug class is found, or a pattern is known, many others start to look for the same types. This in turn leads to waves of similar vulnerability discoveries. This is a prime example of that.

---

[1] A **LPE** is an *unauthorized* way for a unprivileged user to gain **privileged access** (root user) to the system in which these vulnerabilities exists.

[2] A **socket** in Linux and networking is an endpoint for communication between processes, either on the same machine or across a network. It is also a **system call** to transfer execution from a user process into the operating system kernel.

Other times, already reported bugs, that really is vulnerabilities, have stayed unfixed for long time. With the help of the new AI scanner tools, these are *re-discovered*, and there are *PoC code/exploits developed for them*. One good example of is the bug originally reported in 2020[3] by Jann Horn, re-reported by qualys, fixed by linus[4], and a working demo exploit is released[5] that steals SSH keys. The difference here is that there has been a six year period with a bug, with kernel patches, reported by a well-reknown security reasearcher at Google. The probability that someone else (i.e. nation states, APT groups) have developed a working exploit to this, before it now was patched, should be considered medium to high.

4 Some notes on the current analysis of these vulnerabilities

This chapter is to provide the bigger picture, to complement the specifics discussed everywhere else in the document.

We cannot assume that all vulnerable software will be identified, as software is increasingly analyzed for vulnerabilities more advanced vulnerability analysis systems. The new AI based tools are good at discovering bugs that it can identify based on its training, on patterns that it can recognize. There can still be logical errors or bugs that are context-specific. There are still more complex vulnerability classes, or intentionally introduced vulnerabilities[6] that still will be hard to find, even for smarter vulnerability scan software.

Many organisations *underestimate* the problem with the example vulnerabilities since they, according to traditional CVSS[8] scoring, are only considered *high*, not *critical*. The Attack Vector (AV) metric heavily penalizes local access compared to network access, preventing local-only vulnerabilities from reaching Critical (9.0–10.0) under most conditions. CVSS measures exposure characteristics, not strategic value in attack chains. Hence many of the vulnerabilities with high operational impact will not be urgently handled by many organisations, since their strategic value is not properly assessed or evaluated.

The existence of vulnerable code does not automatically imply practical exploitability. Exploitability depends on whether vulnerable functionality is reachable within the constrained execution environment of the affected software component. However, these types of vulnerabilities must be assessed with some additional perspectives:

[3] https://lkml.iu.edu/hypermail/linux/kernel/2010.2/01061.html

[4] https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=31e62c2ebbfdc3fe3dbdf5e02c92a9dc67087a3a

[5] https://github.com/0xdeadbeefnetwork/ssh-keysign-pwn

[6] Sometimes called “bugdoor”, a program error that is a vulnerability intentionally added by the vendor or the programmer

- **Time**: The actual bugs have been in the code for a long period of time. In these cases, around 10 years. There are other recently found vulnerabilities with much longer hidden existence[9]. There are also many examples of when a vulnerability is found and reported, but the vendors may take a very long time to fix it [10], leaving customers exposed during the time it is not publicly known and mitigated. Thus they are latent vulnerabilities that may have existed and potentially been privately exploited for many years (i.e. as zero days) by nation state threat actors or other technically advanced adversaries.
- **Abundance**: Many of these bugs are in the operating system kernel. The Linux kernel is software that exists on billions of devices; there are enterprise servers as well as OT equipment and embedded devices running Linux with kernel versions that have been released during the actual time period.
- **Attack surface**: Typical for many of these new vulnerabilities are that they are very low-level kernel vulnerabilities, in places such as the memory management. Getting any unauthorized or less restricted or unrestricted access to kernel functionality often gives an attacker an significant advantage over the defender.
- **Attack methodology**: Today many attacks are performed as *attack chains*, where different vulnerabilities are used for various steps in a *cyber kill chain*. Another vulnerability might be used to get remote (low level) access into a device. These categories of vulnerabilities are then used to gain unrestricted access inside the device.
- **Skeleton keys**: In modern attack chains, vulnerabilities of this class become highly reusable post-exploitation capabilities that may be combined with many different initial access techniques. These vulnerabilities function as reusable post-exploitation primitives that may be combined with many different initial access techniques. This makes them strategically valuable in modern attack chains, even when the vulnerabilities themselves are classified as local. Several modern exploitation techniques are memory-resident and may leave limited forensic traces on disk.

There are also other bugs that are currently being uncovered that have existed in other main software packages, such as the Apache web server. These might also not be critical, but also – similarly to the perspectives listed above, should be considered in a larger context, before dismissing them as not important to remediate immediately.

When one bug class affects servers, kubernetes clusters and containers, OT, appliances, embedded systems, cloud, virtualization simultaneously, this is something that must be considered systemic cyber risk. These additional perspectives, presented in this document, might give security professionals at an organisation a more realistic understanding that these are severe vulnerabilities. These vulnerabilities should therefore be treated accordingly in well-informed risk assessments and remediation prioritization.

Sysctl prioritizes vulnerabilities based not only on CVSS severity, but also on exploitability, reachability, attack-chain value, affected deployment contexts, and potential systemic impact.

5 Sysctls security stance

Sysctl does not believe that any systems or major applications exist without vulnerabilities. Large software codebases, like the Linux kernel with all its different subsystems, are likely to have many previously undiscovered bugs, some of which have a security impact. We also believe that there are, and likely always will be, actors with knowledge of vulnerabilities, for which there is no fix, and that they have the will and methods to exploit them.

Sysctl's hardening philosophy assumes that vulnerabilities and partial compromises will occur, i.e. the *Assume breach*[7] philosopy, and therefore focuses on containment, attack surface reduction and layered protections.

To get as good as possible protection as well as security stance, Sysctl works both proactively (hardening, defensive coding, etc) as well as reactive (patching when there are new modules available)

Since the beginning Sysctl has used *hardening*[8], and *defense-in-depth*[9] as the foundation of the protection built into the products that we offer. As a result, all programs shipped by Sysctl are configured using multiple security subsystems, to reduce the attack surface exposed to running programs. The purpose of hardening is to introduce additional security boundaries between applications, services, users and sensitive kernel functionality. We have been hardening and using many built-in security mechanisms, such as *Secure Boot*[14], *seccomp*[15], *SELinux*[16], *landlock*[17], etc. We also harden the *kernel configuration*, *kernel module* loading, *systemd* configuration and hardening of many applications. We continously adjust, add configuration and improve the hardening of these. We are in the process of implementing *IMA*[18], working with *remote attestation*[19] and other similar advanced concepts of both **detection** and **protection**.

5.1 Existing mitigations

All of the above vulnerabilities are substantially mitigated by such techniques, mostly by restrictions enforced by *SELinux*, but in some cases also by restrictions imposed by *systemd*. This can also be verified by customers by inspecting the policy[5,6,7].

This does not mean that vulnerabilities cease to exist, nor that they cannot be exploited. It does mean that the vulnerable functionality cannot be reached from the constrained

[7] **Assume Breach** is a proactive strategy and operating model that accepts that attackers will find a way in and focuses on minimizing impact through detection, containment, and recovery.

[8] **System hardening** is the process of reducing a system's attack surface by securely configuring, restricting, disabling, and protecting software, services, accounts, and settings to minimize vulnerabilities and unauthorized access.

[9] **Defense-in-depth** is a security strategy that uses multiple independent layers of protection so that if one security control fails, others still help prevent, detect, or mitigate an attack

execution environments used by Sysctl software, as installed by us in our default install. That is, they operate under restrictive SELinux policies and with specific restrictions to their execution environments.

5.2 Exception to the mitigation rule

The primary purpose of these hardening controls is to constrain application services and non-interactive execution environments. Interactive administrative sessions necessarily require broader system access in order to perform legitimate administrative tasks. For example, the interactive root user or an interactive account added by the customer. If all hardening and mitigation were also applied to interactive accounts it would be impossible to perform several of the tasks that administrative or interactive user account are expected to perform. Therefore, evaluating an exploit from an unrestricted interactive shell does not accurately represent the security posture of hardened services operating under constrained policies.

6 Trends and the current and future threat landscape

As seen by the many examples from the latest waves of vulnerability discoveries and attacks against software, Sysctl believes that the rapid adoption of AI technology has empowered security researchers, criminals, threat actors as well as ordinary developers to analyze source code as well as executables.

With better AI driven tools, reverse engineering and vulnerability research are going to be fundamentally changed. These developments will significantly affect both vulnerability research and defensive operations. More imporantly, these changes impact the speed at which large software ecosystems can now be analyzed and new methodologies, new bug classes and new patterns that can be discovered are turned into vulnerabilities. Once a vulnerability pattern or bug class becomes known, AI-assisted analysis can rapidly identify structurally similar vulnerabilities across very large code bases. All in all, more bugs are going to be uncovered in many software packages over a shorter period of time. This will be problematic for any organisation with many large IT environments. For defenders and operators, there is an increasing need to properly handle the consequences of their discoveries, to avoid being compromised.

Another important consideration in this context is that not all uncovered vulnerabilities will be handled with “responsible disclosure” or “coordinated disclosure”[13]. There will be many more vulnerabilities that will be available, in one way or another – as information or proof-of-concept exploits, before there will be official patches or upgrades from the vendor. Hence, there will be a hard time for all defenders that only rely on patching for their defense.

Besides quicker response time to apply fixes and upgrades, better proactive handling, like Sysctl’s hardening approach, minimizing attack surface, having multiple layers of defense and reducing blast radius/impact of exploited vulnerabilities, are important steps to handle the new threat landscape.

7 Conclusion

Our conclusion can be summarized as this:

- vulnerabilities are inevitable,
- exploitation capability is accelerating,
- layered hardening matters,
- proactive containment is essential,
- local vulnerabilities must not be underestimated,
- reachability and attack surface reduction are key.

Our objective is not to create systems without vulnerabilities, but systems where vulnerabilities are difficult to reach, difficult to exploit, difficult to chain, and limited in impact. The purpose of these controls is not only prevention, but also detection and maintaining operational resilience and containment even under partial compromise scenarios.

8 Contact information

Sysctl AB
Upplandsgatan 88
Stockholm
Sweden

Web: https://www.sysctl.se
Mail: info@sysctl.se
Phone: +46708-330378

9 References

1. https://copy.fail/#exploit
2. https://man7.org/linux/man-pages/man7/address_families.7.html
3. https://github.com/V4bel/dirtyfrag
4. https://github.com/0xdeadbeefnetwork/Copy_Fail2-Electric_Boogaloo
5. Linux command *sesearch --allow -c rxrpc_socket -p create*
6. Linux command *sesearch --allow -c netlink_xfrm_socket -p create*
7. Linux command *sesearch --allow -c alg_socket -p create*
8. *CVSS, https://nvd.nist.gov/vuln-metrics/cvss*
9. https://venturebeat.com/security/mythos-detection-ceiling-security-teams-new-playbook
10. Nokia took 4 years to fix a root exploit. https://www.cve.org/CVERecord?id=CVE-2022-45899
11. https://www.bynar.io/blog/discovery-validation-in-the-linux-kernel-part-1-can-use-after-free-race
12. https://www.wiz.io/vulnerability-database/cve/cve-2025-21896
13. https://en.wikipedia.org/wiki/Coordinated_vulnerability_disclosure
14. Secure Boot https://fedoraproject.org/wiki/Secureboot
15. Seccomp https://en.wikipedia.org/wiki/Seccomp
16. SELinux https://www.redhat.com/en/topics/linux/what-is-selinux
17. Landlock https://landlock.io/
18. IMA https://www.redhat.com/en/blog/how-use-linux-kernels-integrity-measurement-architecture
19. Remote attestation https://www.redhat.com/en/blog/attestation-confidential-computing
20. Fragnesia https://github.com/v12-security/pocs/tree/main/fragnesia
21.