Report

Powered by

TuxCare

A Look at 3 Months of Linux Kernel CVEs

Contents

03 Abstract

04 Target Audience

04 A Note on Kernel Versions

11 Introduction

- 11 What Is a CVE Numbering Authority (CNA)?
- 11 Why Do Organizations Become CNAs?

06 The Linux Kernel as a CNA

08 Everything Is a Potential Security Problem

09 Oddities

09 By the numbers

11 Version by Version

- 11 LTS Kernel 4.4
- 12 LTS Kernel 4.9
- 13 LTS Kernel 4.14
- 14 LTS Kernel 4.19
- 15 LTS Kernel 5.4
- 16 LTS Kernel 5.10
- 17 LTS Kernel 5.15
- 18 LTS Kernel 6.1
- 19 LTS Kernel 6.6

20 Conclusion

Abstract

After many years of gently rebuffing the idea of becoming a CVE Numbering Authority (CNA), the Linux Kernel Project has attained the status of CNA – granting it the power to independently issue Common Vulnerabilities and Exposures (CVEs). In the first three months of its CNA status, the number of CVEs being issued has surged dramatically. The Linux Kernel team has been generating thousands of CVE identifiers, greatly surpassing the conventional rate of CVE assignments, but the CVEs issued by the Kernel team have mostly been for trivial bug fixes and not solely security flaws. Just in May 2024 alone, the Linux team issued over 1,100 CVEs for kernel-related bugs. This flood of CVEs caused by the Kernel team's approach, while garnering support from some corners of the infosec community, has spurred a considerable amount of criticism, even being called a "disaster" by industry thought leaders. In this report, we examine the role of CNAs, the Linux Kernel Project's highly-scrutinized approach to issuing large numbers of CVEs, and what their approach means for enterprise Linux users.

Target Audience

This report is intended to assist Linux system administrators, security researchers, developers, and interested end-users in understanding the fundamental shift in policy regarding the assignment of CVEs to kernel bugs. This could help, for example, in planning and implementing a patching strategy, or in investigating alternatives to current patching practices, that can sustain the increase in potential security problems – while at the same time providing insights into how specific enterprise-focused Linux distributions are susceptible to these new CVEs.

A Note on Kernel Versions

Throughout this report there will be multiple references to various different Linux kernel versions. The Kernel version is in the form <Major>.<Minor>.<Patch>, for example, "6.1.0." The <Major> component of the version is an arbitrary number, increased at the discretion of the Linux "Benevolent Dictator for Life," Linux Torvalds – or as he framed the question in a post on the now defunct Google+ platform, [it would change the major version as] "I'm once more close to running out of fingers and toes"[1] [with which to count the current version numbers, referring to version 3.20 moving to 4.0].

While, for most software packages, <Major> versions indicate API breaking changes, <Minor> indicates non-breaking changes, and <Patch> indicates bug fixes (security or otherwise), the arbitrary nature of the kernel version numbering scheme does not necessarily follow this rule.

Additionally, this report focuses specifically on so-called LTS kernel versions. These are kernel versions that the team behind the kernel commits to maintain in the long run (typically years), versus the non-LTS versions that are supported until the next release, plus a few months. LTS versions are the primary choice for Linux distribution vendors, more so for enterprise-focused Linux distributions.

LTS Versions[2]

3.0	3.2	3.4
3.10	3.12	3.14
3.16	3.18	4.1
4.4	4.9	4.14
4.18	4.19	5.4
5.10	5.15	6.1
6.6		

Also, when considering the number of CVEs discussed in this report and how susceptible different kernel versions are to the vulnerabilities those CVEs refer to, it's important to keep in mind that the same CVE can, and usually does, affect ranges of versions rather than just one specific version.

As will be discussed in subsequent chapters, the Kernel-CNA-assigned CVEs are not assigned to a given version of the kernel[3], but rather offer "hints" towards those versions affected by the vulnerabilities. This is an important enough point to mention, as it was specifically mentioned, repeatedly, by the Kernel CNA team. All the charts and data shown in this report are drawn from the version numbers hinted in each CVE. This process will be described in further detail in other sections of this report.

1 **Linux Version Dilemma: Linus Torvalds is "Running Out of Fingers and Toes"**
https://www.linux.com/news/linux-version-dilemma-linus-torvalds-running-out-fingers-and-toes/

2 **Wikipedia: Linux kernel version history**
https://en.wikipedia.org/wiki/Linux_kernel_version_history

3 https://lore.kernel.org/linux-cve-announce/2024041738-CVE-2024-26920-a681@gregkh/T/

Introduction

As of February 13, 2024, the Linux Kernel Project (kernel.org) is officially a CVE Numbering Authority (CNA), granting it the ability to issue Common Vulnerabilities and Exposures (CVEs) for any vulnerabilities in the Linux kernel as listed on kernel.org – not including end-of-life (EOL) versions. This transformation has resulted in a massive surge of CVEs being issued, which has many in the infosec community expressing their concern.

However, before diving into what this surge means for the Linux community, let's explore what being a CNA has conventionally meant in the Linux world and how organizations typically become CNAs.

What Is a CVE Numbering Authority (CNA)?

CNAs are organizations that are authorized to designate CVE Identifiers (CVE IDs) and publish CVE Records for vulnerabilities impacting products inside their specific scope. These CVE IDs allow the security community to easily get comprehensive details about specific vulnerabilities, enabling effective communication and collaboration in handling security threats. Organizations that gain CNA status accept a significant level of responsibility, demonstrating their vital role in the world of cybersecurity.

CVE IDs for many open-source projects are created and managed by third-party CNAs like GitHub, Red Hat, and MITRE. For a lot of projects, this satisfies the requirements for handling a project's CVE IDs and means that maintainers don't have to channel resources toward managing their own CVE IDs or records.

As of the time of writing, there are 361 CNAs hailing from 40 countries (with a single CNA that has no country affiliation) participating in the CVE Program. The Linux Kernel Project became the 193rd[1] CNA from the United States.

Why Do Organizations Become CNAs?

For open-source projects whose needs have grown beyond what is possible when relying on existing CNAs, becoming a CNA themselves might be a solution. There are several benefits associated with becoming a CNA:

- **Being able to provide high-quality CVE records for users of a project**

  CVE records generated by third parties can be inaccurate or incomplete, so taking on the job themselves offers the capability of delivering more in-depth and precise information.

- **Encouraging researchers to reveal vulnerabilities to the project before being assigned a CVE ID**

  CVEs cannot be assigned for projects in a CNA's particular scope without the CNA first receiving a report. This means that researchers have to first engage with the CNA, minimizing confusion and enabling security subject-matter experts on the project to give their opinion on whether or not to assign a CVE for a specific disclosure. This was a particularly important concern for the Kernel team, as some of the CVEs filed against the Kernel, in the past, turned out not to be security problems after all.

1 **Kernel.org Added as CVE Numbering Authority (CNA)**
https://www.cve.org/Media/News/item/news/2024/02/13/kernel-org-Added-as-CNA

- **Having the ability to assign CVEs without needing to disclose embargoed information to other organizations**

  This enables the project to decide who, if anyone, requires or receives pre-disclosure information.

While hundreds of organizations across 40 countries have decided to enjoy the benefits listed above by becoming a CNA, the Linux Kernel Project has – for the most part – opted not to participate[1] in the CVE ID issuing world...

...until February 13, 2024, that is.

The Linux Kernel as a CNA

As of February 13, 2024, the Linux Kernel Project is officially a CNA, and has since been enjoying the exclusive rights to issue CVE identifiers for the Linux kernel.

However, the decision for the Kernel team to begin issuing CVEs goes directly against the previous stance the team held. Just a few years prior, The Linux Foundation's Greg Kroah-Hartman balked[2] at the idea of becoming a CNA: "The Kernel Community is not a CNA. I don't want to be a CNA."

Before the effective Kernel CNA was established, the Kernel team held the view that security problems would be fixed without a CVE being assigned. In fact, they strongly opposed the idea that CVEs were even beneficial in the first place.

So, why did the Kernel team change its stance? According to the documentation[3] file included with the Linux Kernel source code, where the team provides details on its new CNA status, there is a combination of reasons why they began to seek control over CVE number assignments:

Common Vulnerabilities and Exposure (CVE) numbers were developed as an unambiguous way to identify, define, and catalog publically disclosed security vulnerabilities. Over time, their usefulness has declined with regards to the kernel project, and CVE numbers were very often assigned in inappropriate ways and for inappropriate reasons. Because of this, the kernel development community has tended to avoid them. However, the combination of continuing pressure to assign CVEs and other forms of security identifiers, and ongoing abuses by community members outside of the kernel community has made it clear that the kernel community should have control over those assignments.

1 & 3 **The Linux Kernel Documentation**
https://docs.kernel.org/process/cve.html

2 **Kernel Recipes: "CVEs are dead, long live the CVE!"**
https://kernel-recipes.org/en/2019/talks/cves-are-dead-long-live-the-cve/

Before this decision, the Kernel team would include security patches in updates without getting into the weeds of identifying and numbering each one. The expected, and advised, course of action prescribed by the Kernel team for developers, distribution maintainers, and interested end-users was that they should always run the latest and most up-to-date version of the kernel available – as security fixes were constantly, but silently, introduced in every new release.

During this pre-CNA period, the Kernel team was putting roughly 22 bug fixes[1] into the stable trees per day, which represents about 5% of upstream commits. At the time, this was considered low. At that rate, the Kernel team was putting out one or two stable releases per week.

As such, even without knowing specifically for which threat or for what scenario, running the latest version would result in a more secure kernel, as users would simply enjoy the ~22 patches without really needing to know the specifics of each individual security fix, and in fact, which, if any, of those 22 patches were security related or fixes for run-of-the-mill bugs.

With the introduction of the Kernel CNA, however, this policy had been turned on its head. Now, as laid out in the Kernel CNA documentation, every bug(fix) will be assigned a CVE[2]. The reasoning being that the Kernel team does not know the environment where the software will be used, so they can't reliably predict which bugs may or may not lead to security issues, so – rather than choosing – they are erring on the side of (abundant) caution.

Pros and Cons: The Linux Kernel Becoming a CNA

Pros	Cons
Disclosing More Information With the team that knows the kernel best cataloging all fixes, security or otherwise, Linux users are getting more data about kernel CVEs than they ever have before.	**Making Some Users Especially Vulnerable** With more CVEs, which may or may not turn out to be exploited, now being assigned IDs than ever before, users running versions affected by some of these now public CVEs have a target on their back.
Providing More Transparency With the Kernel Project's new approach, fixes that were once concealed with stable releases are now being brought to light.	**Increased Urgency** As a flood of new CVE IDs are hitting the infosec space, teams are needing to accelerate how quickly they handle them.
CVE IDs Can Be More Easily Revoked If a CVE is not confirmed to be an authentic security flaw, then security and vulnerability management firms can register a CVE revocation request with the precise Kernel team that handles the impacted component.	**Varying Severity Ratings** The Kernel CNA only assigns CVE IDs, but not CVSS scores. Meanwhile, other organizations end up assigning different CVSS scores to the same CVE created by the Kernel team.
No CVEs for Zero-Day Vulnerabilities The Linux Kernel team will also opt to not assign CVEs until a patch has been released, which means there will be no CVEs for zero-days or vulnerabilities that may necessitate a longer reporting and patching lifecycle. This is a good thing as long as you're always running the latest version possible to have those patches included.	**The Methodology Used** CVEs as information are subject to change by reviewers – like code submissions to the Kernel, the CNA takes the same approach in accepting changes to CVE information as "pull requests." Each CVE is disclosed in a publicly accessible repository and openly communicated in an open mailing list. It is not uncommon for comments from other people to be accepted and change the scope or applicability of vulnerabilities and mitigations.

1 **Kernel Recipes: "CVEs are dead, long live the CVE!"**
https://kernel-recipes.org/en/2019/talks/cves-are-dead-long-live-the-cve/

2 **[PATCH v3] Documentation: Document the Linux Kernel CVE process**
https://lwn.net/ml/linux-kernel/2024021430-blanching-spotter-c7c8@gregkh/

It's possible to find arguments for and against each of these positions. On the one hand, more information enables decision makers to decide with more certainty, for example, when deciding on what to patch or what kernel version to prioritize. On the other hand, this leads, and has led, to an explosion in the number of CVEs being created – which we will get into further on in this report.

Everything Is a Potential Security Problem

With the Kernel CNA assigning CVE IDs to trivial bug fixes and not just security flaws, there has been a massive surge in the number of CVE IDs compared to before the Kernel team gained its CNA status.

Consider that **between 2006 and 2018 there were 1005 CVEs assigned directly affecting the Linux Kernel[1]. In just the first three months of CNA operations, 2039 CVEs were issued by the Kernel CNA**. That's about 19 new CVEs, per day, every single day, affecting the Linux kernel since the CNA started to issue them.

Period	# CVEs impacting Kernel	
2006-2018	1005	Approx 84 per year
Feb 13 2024 - May 31 2024	2039	Approx 19 per day

Of these 2039 CVEs in the analyzed period, 60 have been later rejected (approx. 3%). Note that these still required analysis and review to reach this rejected state.

This massive increase in the number of CVEs has a ripple effect in the Linux space and, in particular, on the cybersecurity industry. IT has been relying heavily on CVEs as a measure of risk and a measurable component of a cybersecurity posture – which has been argued may not be the best approach, but it is what we have – and most tools used to scan and manage vulnerabilities rely (almost) exclusively on CVE data to operate.

Another aspect of the CVE issuance process is that the Kernel CNA does not score each CVE. Since risk can be considered subjective and environment dependent, the CVE scoring process has always led to situations where different registries would show risk values differently than others, and, in turn, specific vendors could have different values on top of that. The Kernel CNA sidesteps the problem by not assigning any CVSS score at all. This makes it necessary for third parties to analyze and score each new CVE, emphasizing the need for an even more transparent and independent review process.

In an unfortunate turn of events, just as the Kernel CNA is ramping up CVE issuance, the National Vulnerability Database is slowing down[2] the analysis and enrichment of the new CVEs. This has led to a situation where the majority of the Kernel-CNA-issued CVEs are yet to be (properly) analyzed and scored, which means that a substantial number of vulnerabilities have been disclosed publicly, but for which distribution vendors, end-users, and even developers have yet to grasp the full reach of.

1 **Kernel Recipes 2019 - CVEs are dead, long live the CVE!**
https://docs.kernel.org/process/cve.html

2 **NIST's National Vulnerability Database Put CVE Enrichment on Hold**
https://heimdalsecurity.com/blog/nvd-stopped-updating-cves/

Oddities

When a CVE is requested from a CNA, it will be assigned an identifier in the form "CVE-YEAR-NUMBER," where "YEAR" is the current year, as of the date of the request. This provides an at-a-glance indication of the time period in which the vulnerability was disclosed. With that in mind, one aspect that stands out when looking at the list of issued CVEs for the Kernel CNA is the very large number of CVEs with year indication from previous years, including entries from 2019 and 2020, or more than four years ago. Since the CVEs only started to be issued in 2024, this can introduce confusion if you're not familiar with the process and are simply reading a compliance report or a vulnerability scanner's output.

This "backfilling" practice by the Kernel CNA departs from other CNAs' operating processes. The year in the identifier appears to be related to the associated bugfix code commit, which, if taken to its most remote possible code submission, can lead to CVEs being requested, issued, and analyzed in 2024, but for which the year part refers to, potentially, decades ago.

Period	2019	2020	2021	2022	2023	2024
# CVEs	3	14	664	75	403	880

CVE distribution per year of identifier, during the first 3 months of Kernel CNA.

Compare this practice with, for example, a bug associated with the "curl" project: CVE-2022-35252[1]. This bug, assigned in 2022, as the identifier clearly states, refers to a bug introduced to the codebase in 1999, and yet the year part of the CVE identifier continues to show the assignment year.

By the numbers

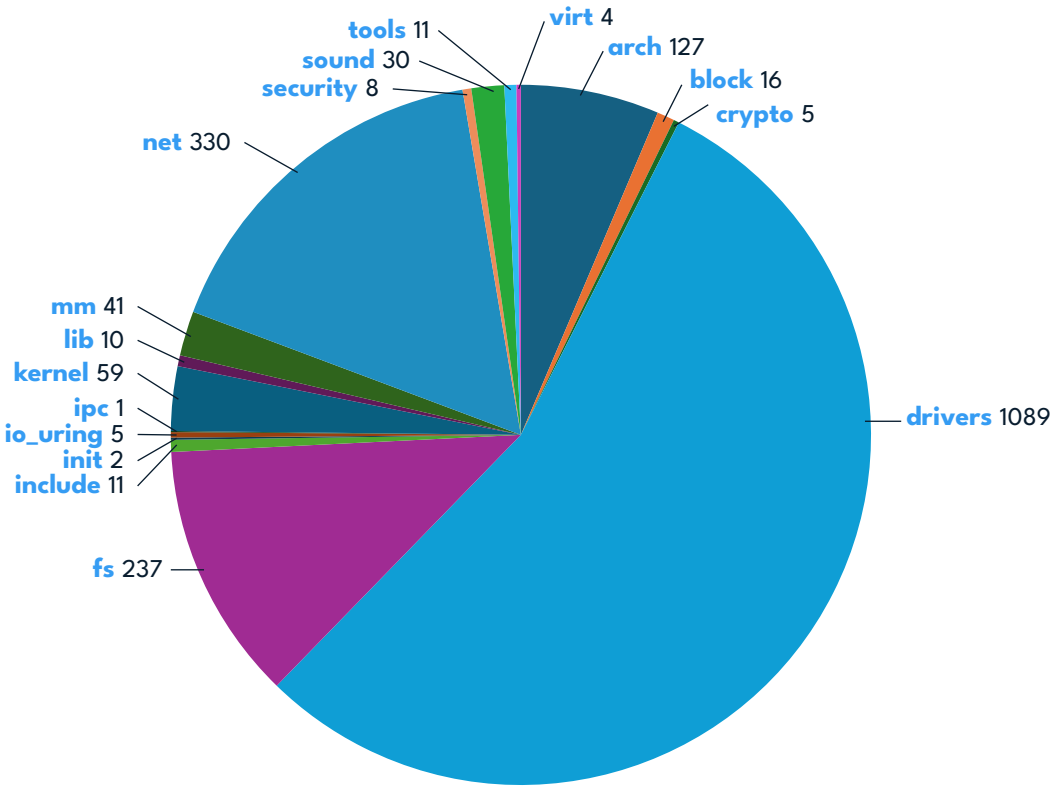
For the period between February 13th and May 31st, below you can see the CVE tally affecting each LTS kernel version. The same CVE can affect multiple versions.

Kernel version 3.16	Affected by 1 CVE	Kernel version 3.18	Affected by 1 CVE
Kernel version 4.4	Affected by 121 CVEs	Kernel version 4.9	Affected by 158 CVEs
Kernel version 4.14	Affected by 236 CVEs	Kernel version 4.19	Affected by 501 CVEs
Kernel version 5.4	Affected by 692 CVEs	Kernel version 5.10	Affected by 1036 CVEs
Kernel version 5.15	Affected by 845 CVEs	Kernel version 6.1	Affected by 841 CVEs
Kernel version 6.6	Affected by 981 CVEs		

1 **A BUG THAT WAS 23 YEARS OLD OR NOT**
https://daniel.haxx.se/blog/2022/09/05/a-bug-that-was-23-years-old-or-not/

When looking at the count of CVEs affecting each LTS version, it's important to consider how long a given version has been in use. While version 5.10 has a higher count than version 6.6, it has been in use for over 42 months. Version 6.6, with an almost similar number of vulnerabilities affecting it, was released only 8 months ago.

Now looking at the impacted subsystems, this is the count for the same period:



Note

"Subsystems" are counted by looking at the field programFiles in the json description for each CVE. Several CVEs affect multiple systems. For example, some CVEs will impact a given subsystem and simultaneously also affect "include" or "Documentation." The following subsystems are not counted when there is another subsystem present for a given CVE: "include," "Documentation," "kernel," "trace." Those subsystems are only counted when a CVE affects just source code files residing inside those particular locations in the kernel tree.

Note

Even with the previous restriction, 7 CVEs are still counted twice, as they impact multiple locations in the kernel, outside the aforementioned exceptions.

Note

At the time the data was compiled, a handful of CVEs still contained some problems in the json data, like having no programFiles field at all (eg: CVE-2023-52653)

Version by Version

LTS Kernel 4.4

Release Date:	Version Range:	Major Enterprise Linux Distributions Using 4.4
January 10, 2016	4.4.0 - 4.4.302	• SUSE Linux Enterprise Server 12 SP2 and later • Ubuntu 16.04 LTS (Xenial Xerus)

Notable Changes and Improvements[1]:

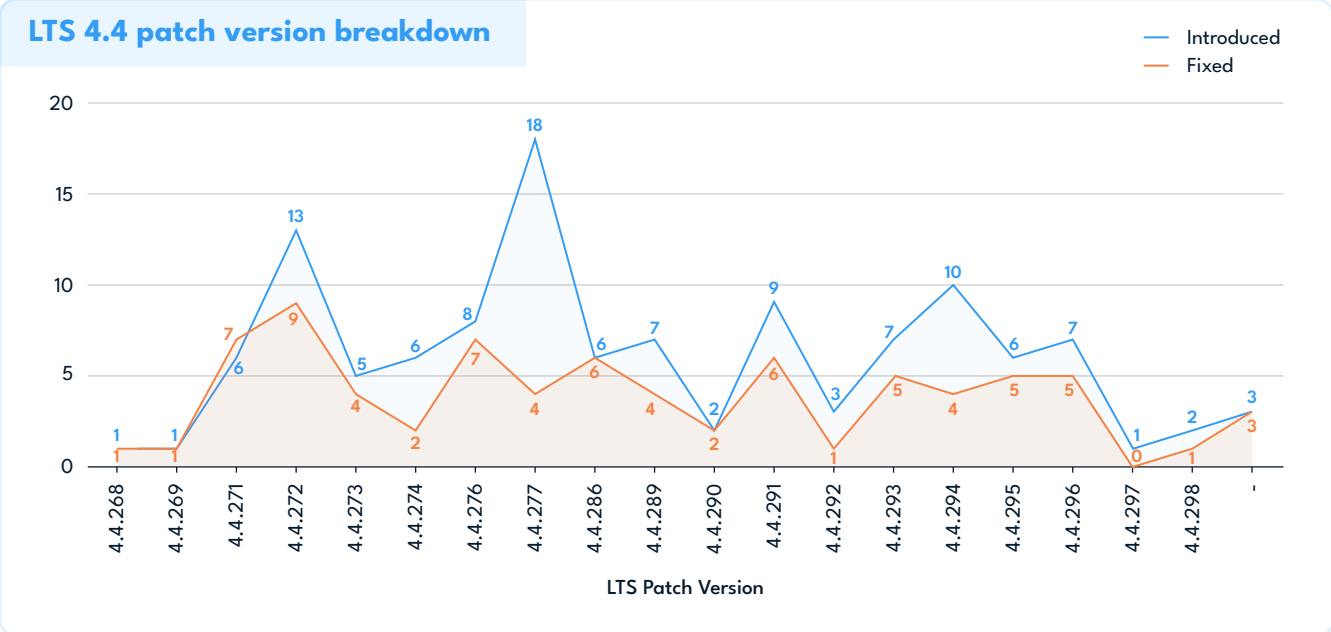
- Direct I/O and Asynchronous I/O support
- TCP listener was made lockless, significantly increasing TCP servers' performance
- eBPF improvements

CVE Information:

The 4.4 series is impacted by 121 of the CVEs disclosed by the Kernel CNA.

The following chart shows the distribution of CVEs introduced and fixed, per patch version, within the 4.4 version range.

This version was already out of official support when the Kernel CNA started to operate.



1 **Kernel Newbies: Linux 4.4**
https://kernelnewbies.org/Linux_4.4

LTS Kernel 4.9

Release Date:	Version Range:	Major Enterprise Linux Distributions Using 4.9
December 11, 2016	4.9.0 - 4.9.337	• Debian 9 (Stretch)

Notable Changes and Improvements[1]:

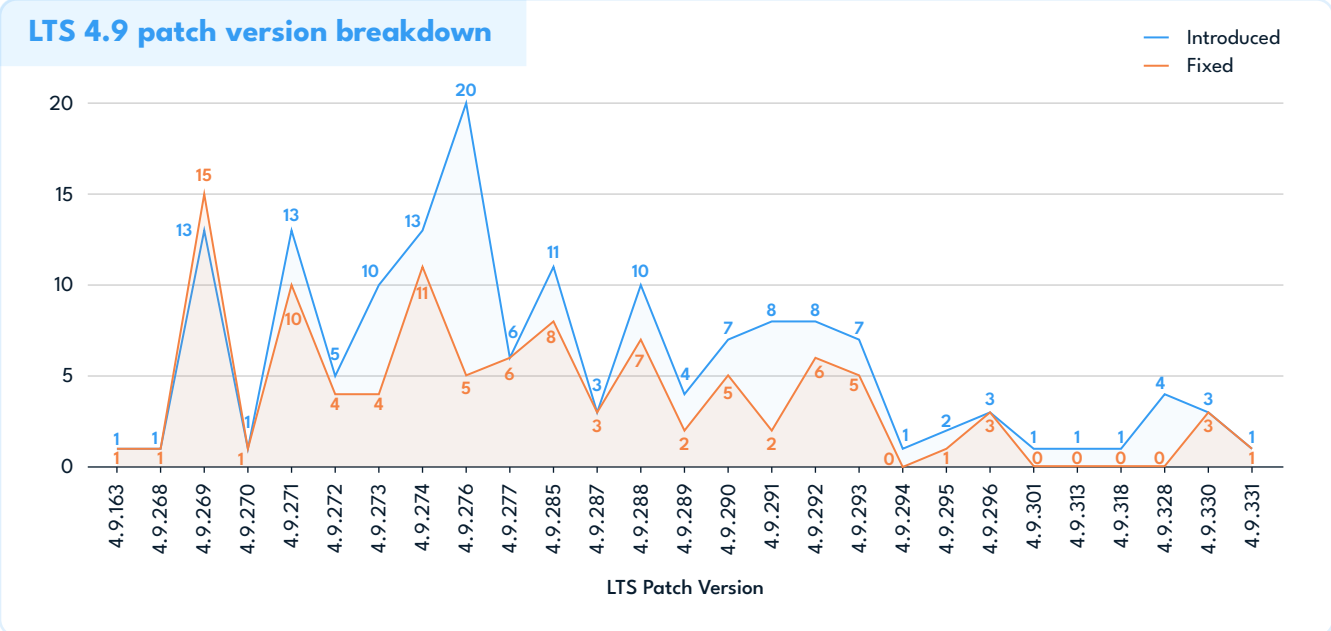
- Improved XFS support
- Hardware latency tracer
- Protection keys support added to through syscalls

CVE Information:

The 4.9 series is impacted by 158 of the CVEs disclosed by the Kernel CNA.

The following chart shows the distribution of CVEs introduced and fixed, per patch version, within the 4.9 version range. CVEs introduced and fixed within the series tend to exist in the initial half of the range, with later versions showing less impact.

This version was already out of official support when the Kernel CNA started to operate.



1 **Kernel Newbies: Linux 4.9**
https://kernelnewbies.org/Linux_4.9

LTS Kernel 4.14

Release Date:	Version Range:	Major Enterprise Linux Distributions Using 4.14
November 12, 2017	4.14.0 - 4.14.336	• Debian 10 (Buster) • SUSE Linux Enterprise Server 15 • CentOS/RHEL 7.6

Notable Changes and Improvements[1]:

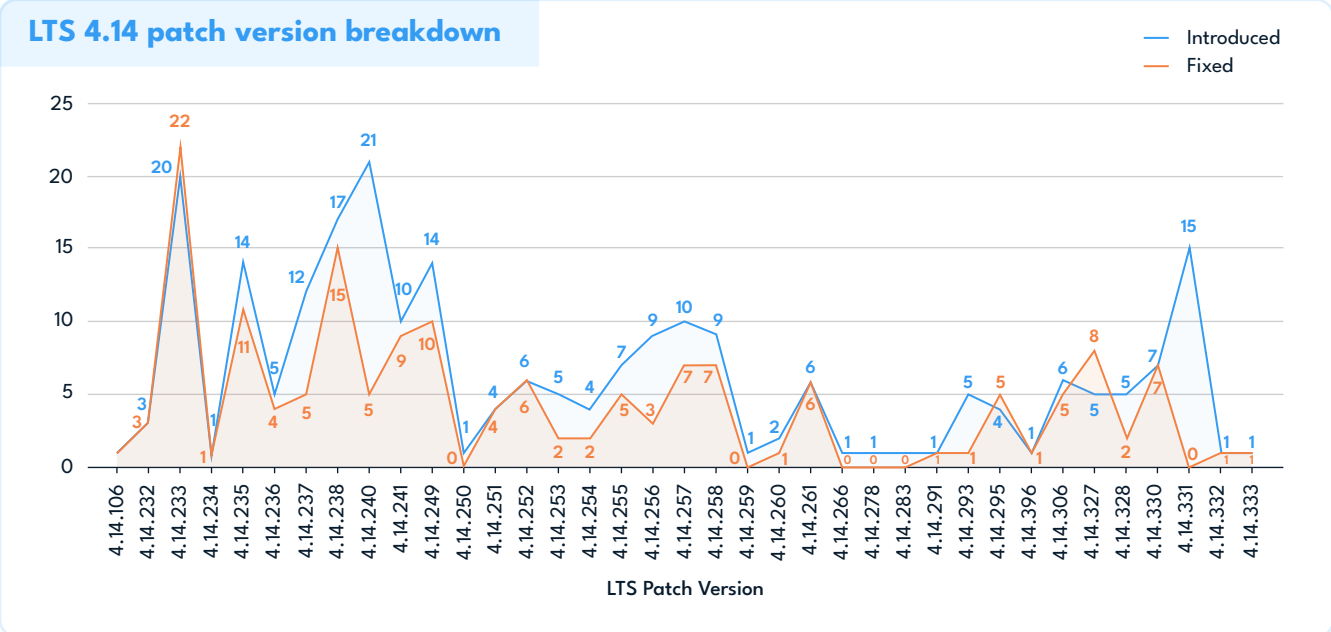
- Increased memory limits (to 128 PiB)
- Support for AMD Secure Memory Encryption
- ORC unwinder for kernel traces
- Zero-copy from user memory to sockets

CVE Information:

The 4.14 series is impacted by 236 of the CVEs disclosed by the Kernel CNA.

The following chart shows the distribution of CVEs introduced and fixed, per patch version, within the 4.14 version range. Since this series is no longer supported, CVE distribution across versions is spread out, with a more pronounced bump earlier in the series.

This version was already out of official support when the Kernel CNA started to operate.



1 **Kernel Newbies: Linux 4.14**
https://kernelnewbies.org/Linux_4.14

LTS Kernel 4.19

Release Date:	Version Range:	Major Enterprise Linux Distributions Using 4.19
October 22, 2018	4.19.0 - 4.19.316 (still supported)	• Debian 10 (Buster) • Ubuntu 18.04.3 LTS (Bionic Beaver) • CentOS/RHEL 8

Notable Changes and Improvements[1]:

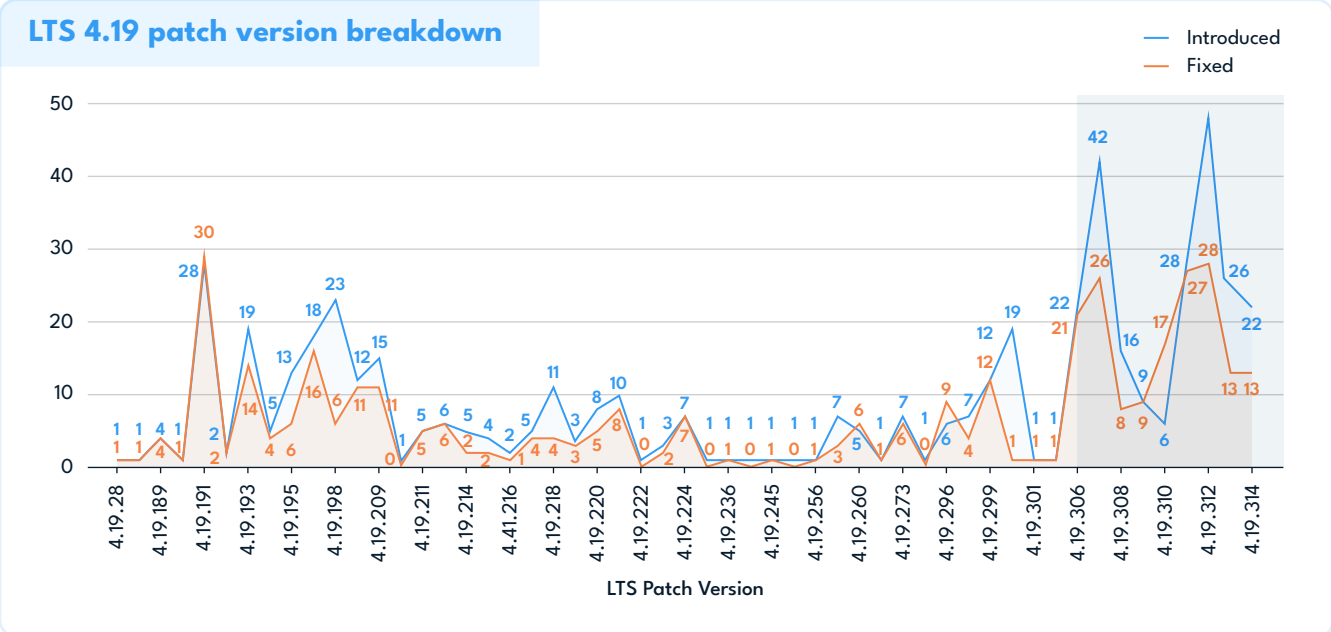
- Introduction of the CAKE (Common Applications Kept Enhanced) network queue management algorithm.
- Multiple CPU security bug fixes (L1TF, Spectre V2, etc)
- cGroup IO latency controller
- New asynchronous I/O polling interface

CVE Information:

The 4.19 series is impacted by 501 of the CVEs disclosed by the Kernel CNA.

The following chart shows the distribution of CVEs introduced and fixed, per patch version, within the 4.19 version range. There is a clear bump in the number of introduced and fixed CVEs, as the series matures.

This LTS range was at patch version ".306" on February 13, when the Kernel CNA started to operate. This coincides with the first major bump in CVE count.



1 **Kernel Newbies: Linux 4.19**
https://kernelnewbies.org/Linux_4.19

LTS Kernel 5.4

Release Date:	Version Range:	Major Enterprise Linux Distributions Using 5.4
November 24, 2019	5.4.0 - 5.4.278 (still supported)	• Ubuntu 20.04 LTS (Focal Fossa) • Amazon Linux 2 • CentOS/RHEL 8.3

Notable Changes and Improvements[1]:

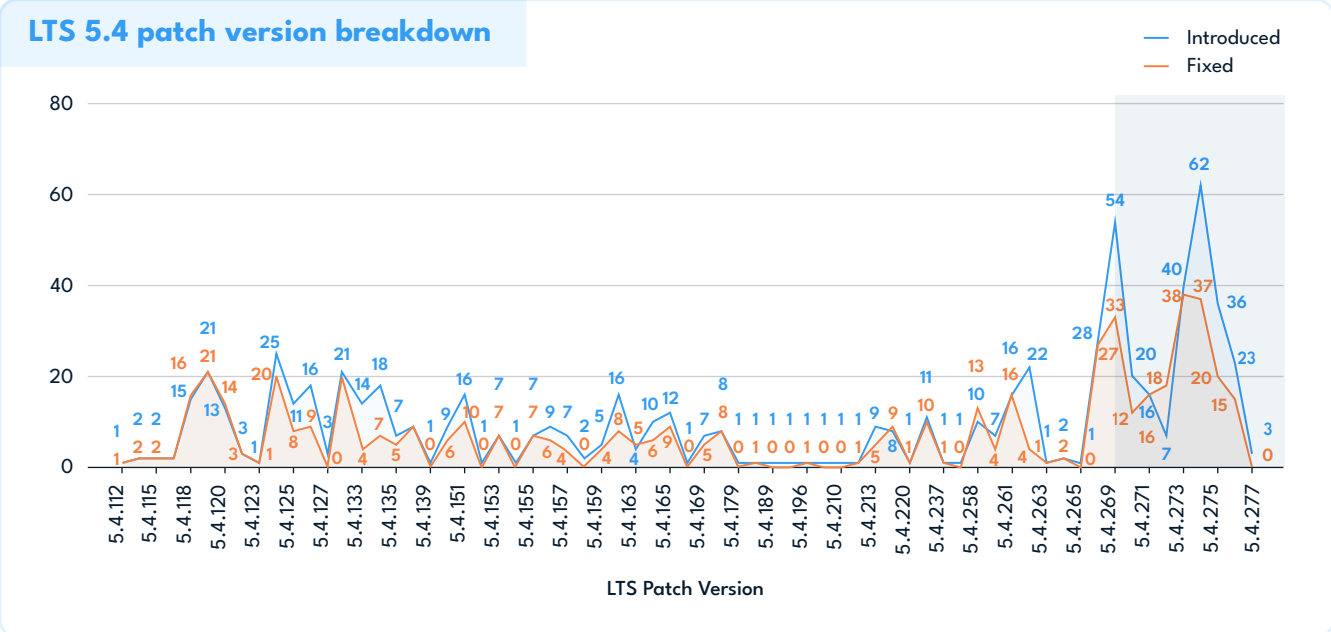
- Introduction of the Lockdown module to enhance kernel security by restricting access to kernel features.
- Inclusion of the fs-verity file integrity mechanism.
- Support for the exFAT file system, enabling better compatibility with Windows file systems.
- Virtio-fs filesystem sharing functionality with virtualized guests

CVE Information:

The 5.4 series is impacted by 692 of the CVEs disclosed by the Kernel CNA.

The following chart shows the distribution of CVEs introduced and fixed, per patch version, within the 5.4 version range. The majority of CVEs are introduced, and fixed, in the later versions of the series.

This LTS range was at patch version ".268" on February 13, when the Kernel CNA started to operate. All versions on the 5.4 series after .268 have considerably higher CVE counts affecting them.



1 **Kernel Newbies: Linux 5.4**
https://kernelnewbies.org/Linux_5.4

LTS Kernel 5.10

Release Date:	Version Range:	Major Enterprise Linux Distributions Using 5.10
December 13, 2020	5.10.0 - 5.10.220 (still supported)	• Debian 11 (Bullseye) • Ubuntu 20.04.3 LTS (Focal Fossa) • CentOS/RHEL 8.4

Notable Changes and Improvements[1]:

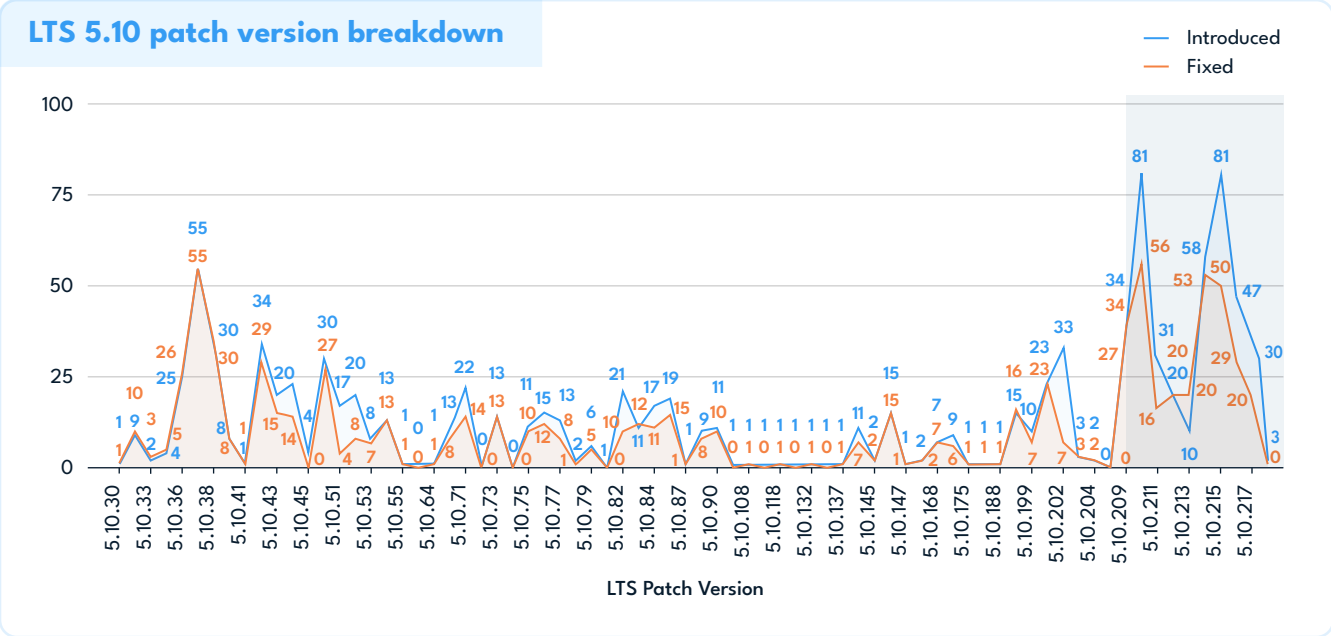
- Io_uring restrictions
- Ext4 performance improvements
- Performance improvement to (partially) mitigate the performance hit caused by Spectre-forced changes

CVE Information:

The 5.10 series is impacted by 1036 of the CVEs disclosed by the Kernel CNA.

The following chart shows the distribution of CVEs introduced and fixed, per patch version, within the 5.10 version range. A significant number of introduced and fixed CVEs is noticeable in the later versions in the series.

This LTS series was at patch version ".209" on February 13, when the Kernel CNA started to operate. This coincided with the increased number of CVEs affecting later versions.



1 **Kernel Newbies: Linux 5.10**
https://kernelnewbies.org/Linux_5.10

LTS Kernel 5.15

Release Date:	Version Range:	Major Enterprise Linux Distributions Using 5.15
October 31, 2021	5.15.0 - 5.15.161 (still supported)	• Ubuntu 22.04 LTS (Jammy Jellyfish) • Fedora 35 • CentOS Stream 9

Notable Changes and Improvements[1]:

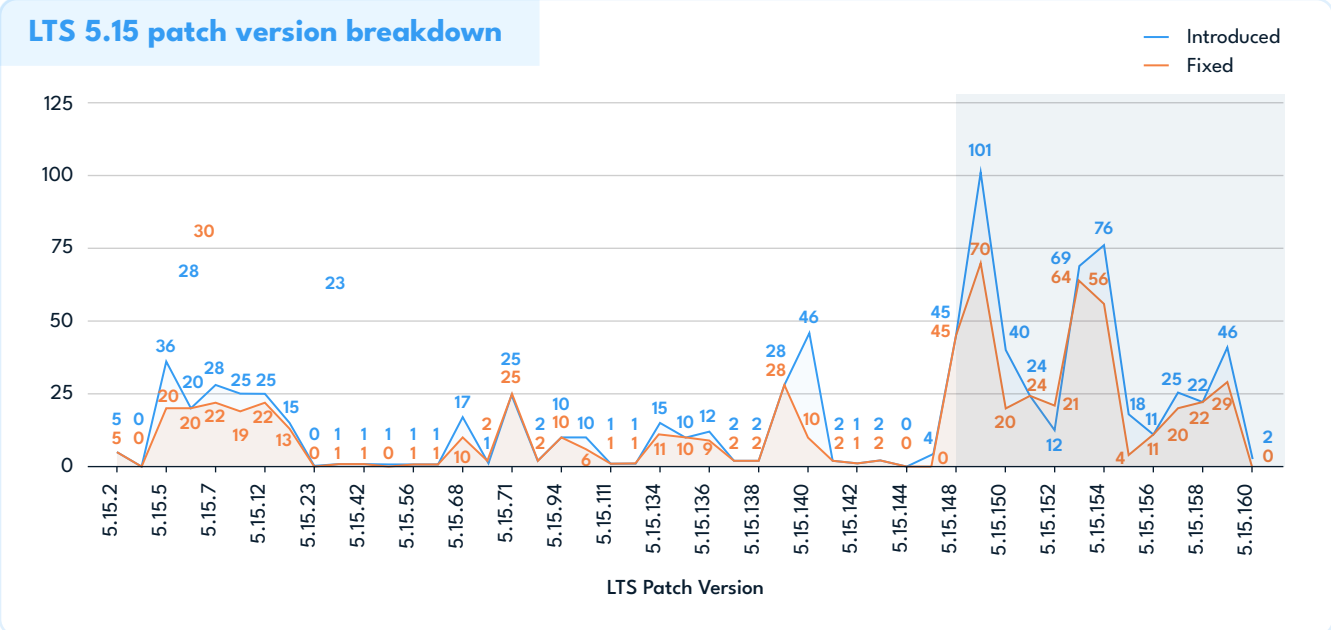
- Introduction of the NTFS3 file system driver, providing better support for NTFS file systems.
- Ksmbd in-kernel SMB 3 server
- Support for the DAMON (Data Access MONitor) subsystem
- Support for asymmetric scheduling affinity

CVE Information:

The 5.15 series is impacted by 845 of the CVEs disclosed by the Kernel CNA.

The following chart shows the distribution of CVEs introduced and fixed, per patch version, within the 5.15 version range. There is a noticeable difference in the amount of introduced and fixed CVEs, with the most recent versions having both more CVEs affecting them, as well as more fixes introduced to address those CVEs.

The 5.15 LTS range was at patch version ".148" on February 13, when the Kernel CNA started to operate. This also coincided with the significant spike in disclosed vulnerabilities impacting subsequent versions in the series.



1 **Kernel Newbies: Linux 5.15**
https://kernelnewbies.org/Linux_5.15

LTS Kernel 6.1

Release Date:	Version Range:	Major Enterprise Linux Distributions Using 6.1
December 11, 2022	6.1.0 - 6.1.95 (still supported)	• Debian 12 (Bookworm) • Fedora 37

Notable Changes and Improvements[1]:

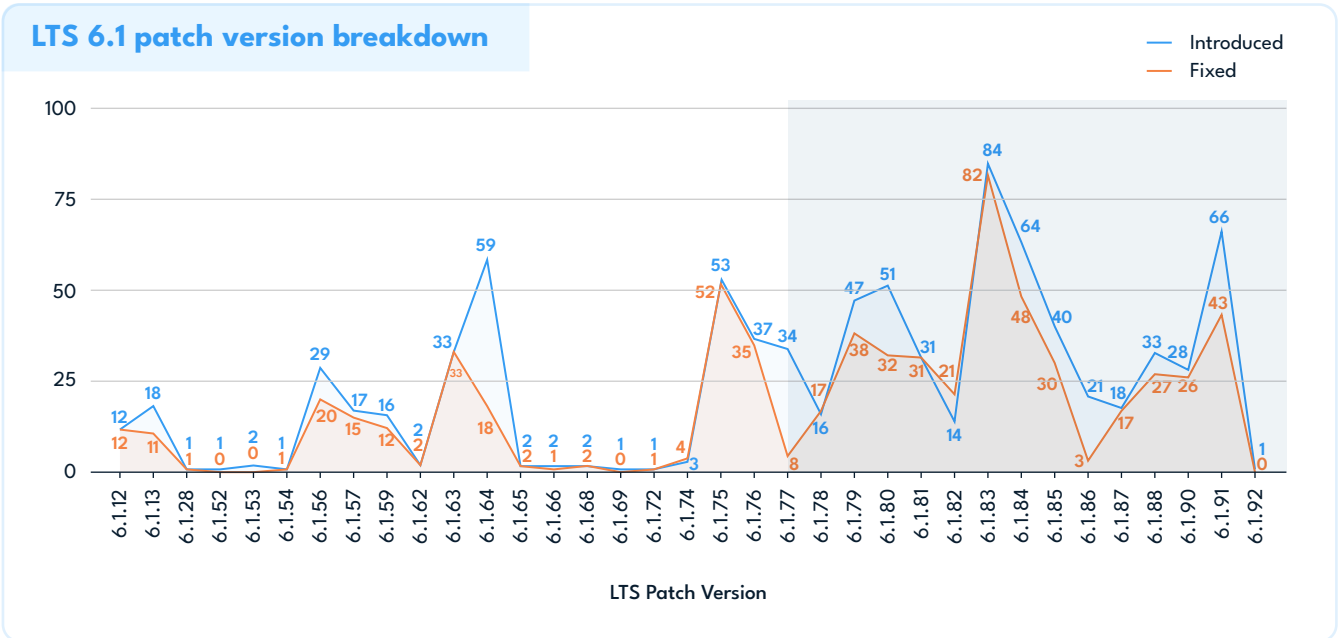
- Introduction of the Rust for Linux infrastructure, enabling the development of kernel modules in Rust.
- Support for the MGLRU (multigenerational LRU) memory management feature.
- Introduction of KMSAN (kernel memory sanitizer)
- Memory tiering improvements

CVE Information:

The 6.1 series is impacted by 841 of the CVEs disclosed by the Kernel CNA.

The following chart shows the distribution of CVEs introduced and fixed, per patch version, within the 6.1 version range. More recent patch versions are showing a larger number of introduced and fixed CVEs.

The 6.1 LTS range was at patch version ".77" on February 13, when the Kernel CNA started to operate.



1 **Kernel Newbies: Linux 6.1**
https://kernelnewbies.org/Linux_6.1

LTS Kernel 6.6

Release Date:	Version Range:	Major Enterprise Linux Distributions Using 6.1
October 30, 2023	6.6.0 - 6.6.35 (still supported)	• Yet to be widely adopted in major enterprise distributions (as of early 2024).

Notable Changes and Improvements[1]:

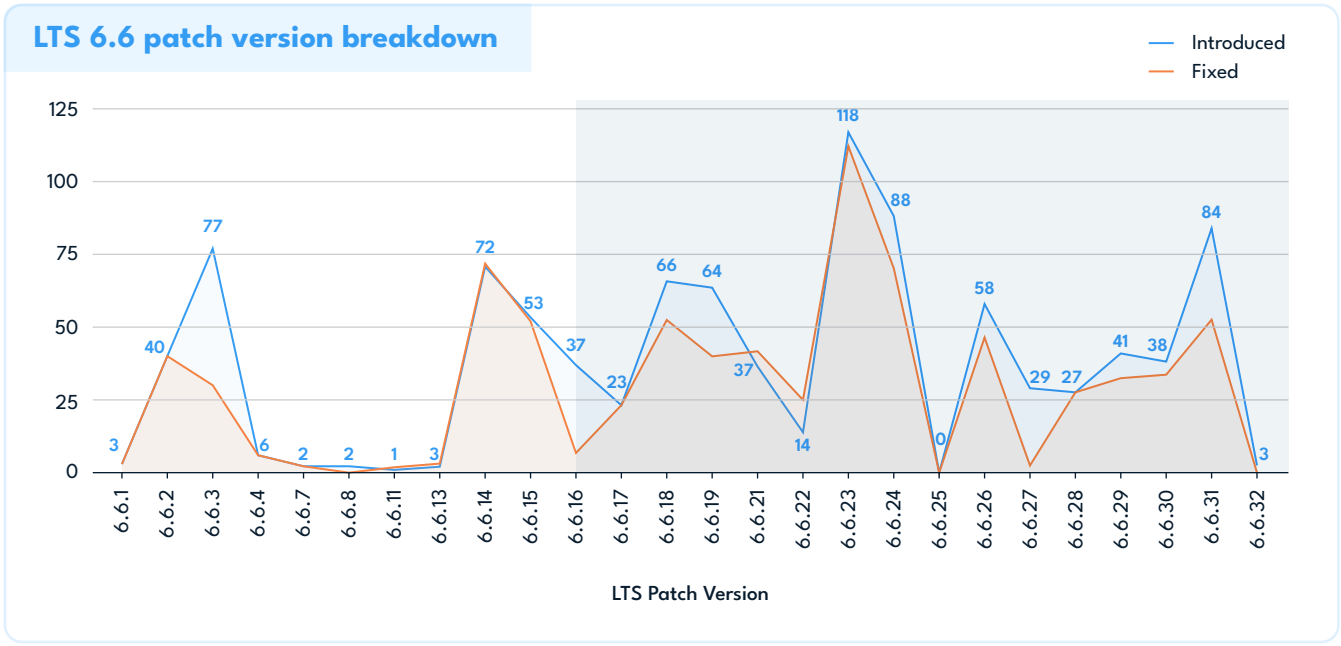
- Faster asynchronous Direct I/O through io_uring
- Shadow stacks security mechanism
- EEVDF task scheduler

CVE Information:

The 6.6 series is impacted by 981 of the CVEs disclosed by the Kernel CNA.

The following chart shows the distribution of CVEs introduced and fixed, per patch version, within the 6.6 version range. Since this is the most current LTS version, it is also the one receiving the most “immature” code - meaning the code that has had less real-world use exposure. This is reflected in the very large numbers of CVEs introduced and fixed in most patch versions.

The 6.6 LTS range was at patch version “.16” on February 13, when the Kernel CNA started to operate. While there are more CVEs issued on subsequent versions, the pattern is not as apparent, likely because this specific range was still very recent. It’s important to note that this was the most-up-to-date LTS version available at the time, so it was the one getting more attention when looking for vulnerabilities.



1 **Kernel Newbies: Linux 6.6**
https://kernelnewbies.org/Linux_6.6

Conclusion

More than simply looking at the numbers, it is important to approach the problem of how to handle such a large increase in new vulnerabilities from an operational and patching point of view. This severe growth in new vulnerabilities introduces difficult challenges for organizations.

First, traditional patching practices, already woefully inadequate in handling problems to meet the current security and regulatory requirements, are once again questioned. With an average of 19 new vulnerabilities impacting the Kernel disclosed every day, there is no maintenance schedule capable of handling this burden without introducing severe business disruption and unsustainable operational costs.

What's more, the current Kernel CNA process is not expected to deviate from the current "every-bug-gets-a-CVE" approach, which means that this large number of vulnerabilities, if anything, is only expected to remain at the current level or grow beyond it, further compounding the problem.

Second, organizations that are frequently updating to the newest mainstream kernel are subjecting themselves to code that likely has more security bugs than previous versions. This is made clear by the data analyzed in this report, which shows that the most recent versions are disproportionately more affected by new flaws than older ones. This is not because people aren't looking for flaws on older versions, it's simply because – by its nature – a new kernel code release takes a few years to mature, at which point many of the bugs that once plagued it have been cleaned up.

There's a wide range of possible explanations for the disparity in vulnerability numbers affecting Kernel versions of different "ages." Vulnerability identification in a given piece of software tends to follow a three-phase process, where each phase corresponds to how widely used it is. Phase 1 corresponds to the release stage, where the piece of software is new, and users start switching to it. This real-world usage pattern attracts regular users and security researchers. In a second phase, the software is widely used and vulnerability discovery will be in full swing. The final phase is when the software begins to be replaced by newer releases, and vulnerability detection and research slows down as the focus shifts away from it. This corresponds with what the data gathered for this report is showing: in phase 3, older Kernel versions are less susceptible to new vulnerabilities and thus more secure[1].

This can be seen, for example, in the fact that kernel version 5.10 is affected by 1036, but has been available for about 42 months (averaging roughly 25 vulnerabilities per month). Kernel version 6.6 is affected by 981 but has been available for only 8 months (for an average of 123 vulnerabilities per month). Interestingly, this holds true if you're comparing the total number of vulnerabilities affecting a kernel series like 5.10, and also within each series, when comparing older (initial) versions in the series with more recent ones.

So, as it's evident that older kernel versions are less inflicted by vulnerabilities and safer to use as a result, what can organizations do?

The good news is that enterprises don't need to constantly update to the newest mainstream Kernel version. Delaying an update to the newest version is a solution for avoiding newer code with higher volumes of bugs, and – fortunately – organizations can do exactly that while still receiving the latest security fixes. Specialized providers are able to deliver solely security fixes to older kernel versions without adding anything else – allowing organizations to enjoy the benefits of relying on more "mature" kernel code without sacrificing security.

1 **Alhazmi, Omar & Malaiya, Yashwant & Ray, Indrajit. (2007). Measuring, analyzing and predicting security vulnerabilities in software systems. Computers & Security. 26. 219-228. 10.1016/j.cose.2006.10.002.**
https://www.researchgate.net/publication/220615003_Measuring_analyzing_and_predicting_security_vulnerabilities_in_software_systems