Search

vendor:linux

Search

Use the Query Builder to create your own search query, or check out the documentation to learn the search syntax.

Search Examples

CVEs in KEV | CVEs with EPSS >= 80% | Crit. Microsoft | High Apache | SQL Injection (CWE-89) | Linux Kernel | High (CVSS 3.1) | Apache Struts | RCE (Remote Code Execution) | XSS (CWE-79) | Critical (CVSS 4.0) | CVEs to check

Search Results (18722 CVEs found)

Export CSV

CVE	Vendors	Products	Updated	CVSS v3.1
CVE-2026-46300	1 Linux	2 Kernel, Linux Kernel	2026-05-25	7.8 High

In the Linux kernel, the following vulnerability has been resolved: net: skbuff: preserve shared-frag marker during coalescing skb_try_coalesce() can attach paged frags from @from to @to. If @from has SKBFL_SHARED_FRAG set, the resulting @to skb can contain the same externally-owned or page-cache-backed frags, but the shared-frag marker is currently lost. That breaks the invariant relied on by later in-place writers. In particular, ESP input checks skb_has_shared_frag() before deciding whether an uncloned nonlinear skb can skip skb_cow_data(). If TCP receive coalescing has moved shared frags into an unmarked skb, ESP can see skb_has_shared_frag() as false and decrypt in place over page-cache backed frags. Propagate SKBFL_SHARED_FRAG when skb_try_coalesce() transfers paged frags. The tailroom copy path does not need the marker because it copies bytes into @to's linear data rather than transferring frag descriptors.

CVE-2026-43503	1 Linux	1 Linux Kernel	2026-05-25	N/A

In the Linux kernel, the following vulnerability has been resolved: net: skbuff: propagate shared-frag marker through frag-transfer helpers Two frag-transfer helpers (__pskb_copy_fclone() and skb_shift()) fail to propagate the SKBFL_SHARED_FRAG bit in skb_shinfo()->flags when moving frags from source to destination. __pskb_copy_fclone() defers the rest of the shinfo metadata to skb_copy_header() after copying frag descriptors, but that helper only carries over gso_{size,segs, type} and never touches skb_shinfo()->flags; skb_shift() moves frag descriptors directly and leaves flags untouched. As a result, the destination skb keeps a reference to the same externally-owned or page-cache-backed pages while reporting skb_has_shared_frag() as false. The mismatch is harmful in any in-place writer that uses skb_has_shared_frag() to decide whether shared pages must be detoured through skb_cow_data(). ESP input is one such writer (esp4.c, esp6.c), and a single nft 'dup to <local>' rule -- or any other nf_dup_ipv4() / xt_TEE caller -- is enough to land a pskb_copy()'d skb in esp_input() with the marker stripped, letting an unprivileged user write into the page cache of a root-owned read-only file via authencesn-ESN stray writes. Set SKBFL_SHARED_FRAG on the destination whenever frag descriptors were actually moved from the source. skb_copy() and skb_copy_expand() share skb_copy_header() too but linearize all paged data into freshly allocated head storage and emerge with nr_frags == 0, so skb_has_shared_frag() returns false on its own; they need no change. The same omission exists in skb_gro_receive() and skb_gro_receive_list(). The former moves the incoming skb's frag descriptors into the accumulator's last sub-skb via two paths (a direct frag-move loop and the head_frag + memcpy path); the latter chains the incoming skb whole onto p's frag_list. Downstream skb_segment() reads only skb_shinfo(p)->flags, and skb_segment_list() reuses each sub-skb's shinfo as the nskb -- both p and lp must carry the marker. The same omission also exists in tcp_clone_payload(), which builds an MTU probe skb by moving frag descriptors from skbs on sk_write_queue into a freshly allocated nskb. The helper falls into the same family and warrants the same fix for consistency; no TCP TX-side in-place writer is currently known to reach a user page through this gap, but a future consumer depending on the marker would regress silently. The same omission exists in skb_segment(): the per-iteration flag merge takes only head_skb's flag, and the inner switch that rebinds frag_skb to list_skb on head_skb-frags exhaustion does not fold the new frag_skb's flag into nskb. Fold frag_skb's flag at both sites so segments drawing frags from frag_list members carry the marker.

CVE-2024-41055	2 Linux, Redhat	4 Linux Kernel, Enterprise Linux, Rhel E4s *and 1 more*	2026-05-23	5.5 Medium

In the Linux kernel, the following vulnerability has been resolved: mm: prevent derefencing NULL ptr in pfn_section_valid() Commit 5ec8e8ea8b77 ("mm/sparsemem: fix race in accessing memory_section->usage") changed pfn_section_valid() to add a READ_ONCE() call around "ms->usage" to fix a race with section_deactivate() where ms->usage can be cleared. The READ_ONCE() call, by itself, is not enough to prevent NULL pointer dereference. We need to check its value before dereferencing it.

CVE-2024-41049	2 Linux, Redhat	3 Linux Kernel, Enterprise Linux, Rhel Eus	2026-05-23	7.0 High

In the Linux kernel, the following vulnerability has been resolved: filelock: fix potential use-after-free in posix_lock_inode Light Hsieh reported a KASAN UAF warning in trace_posix_lock_inode(). The request pointer had been changed earlier to point to a lock entry that was added to the inode's list. However, before the tracepoint could fire, another task raced in and freed that lock. Fix this by moving the tracepoint inside the spinlock, which should ensure that this doesn't happen.

CVE-2024-41022	1 Linux	1 Linux Kernel	2026-05-23	5.5 Medium

In the Linux kernel, the following vulnerability has been resolved: drm/amdgpu: Fix signedness bug in sdma_v4_0_process_trap_irq() The "instance" variable needs to be signed for the error handling to work.

CVE-2024-41018	1 Linux	1 Linux Kernel	2026-05-23	5.5 Medium

In the Linux kernel, the following vulnerability has been resolved: fs/ntfs3: Add a check for attr_names and oatbl Added out-of-bound checking for *ane (ATTR_NAME_ENTRY).

CVE-2024-40916	1 Linux	1 Linux Kernel	2026-05-23	5.5 Medium

In the Linux kernel, the following vulnerability has been resolved: drm/exynos: hdmi: report safe 640x480 mode as a fallback when no EDID found When reading EDID fails and driver reports no modes available, the DRM core adds an artificial 1024x786 mode to the connector. Unfortunately some variants of the Exynos HDMI (like the one in Exynos4 SoCs) are not able to drive such mode, so report a safe 640x480 mode instead of nothing in case of the EDID reading failure. This fixes the following issue observed on Trats2 board since commit 13d5b040363c ("drm/exynos: do not return negative values from .get_modes()"): [drm] Exynos DRM: using 11c00000.fimd device for DMA mapping operations exynos-drm exynos-drm: bound 11c00000.fimd (ops fimd_component_ops) exynos-drm exynos-drm: bound 12c10000.mixer (ops mixer_component_ops) exynos-dsi 11c80000.dsi: [drm:samsung_dsim_host_attach] Attached s6e8aa0 device (lanes:4 bpp:24 mode-flags:0x10b) exynos-drm exynos-drm: bound 11c80000.dsi (ops exynos_dsi_component_ops) exynos-drm exynos-drm: bound 12d00000.hdmi (ops hdmi_component_ops) [drm] Initialized exynos 1.1.0 20180330 for exynos-drm on minor 1 exynos-hdmi 12d00000.hdmi: [drm:hdmiphy_enable.part.0] *ERROR* PLL could not reach steady state panel-samsung-s6e8aa0 11c80000.dsi.0: ID: 0xa2, 0x20, 0x8c exynos-mixer 12c10000.mixer: timeout waiting for VSYNC ------------[cut here]------------ WARNING: CPU: 1 PID: 11 at drivers/gpu/drm/drm_atomic_helper.c:1682 drm_atomic_helper_wait_for_vblanks.part.0+0x2b0/0x2b8 [CRTC:70:crtc-1] vblank wait timed out Modules linked in: CPU: 1 PID: 11 Comm: kworker/u16:0 Not tainted 6.9.0-rc5-next-20240424 #14913 Hardware name: Samsung Exynos (Flattened Device Tree) Workqueue: events_unbound deferred_probe_work_func Call trace: unwind_backtrace from show_stack+0x10/0x14 show_stack from dump_stack_lvl+0x68/0x88 dump_stack_lvl from __warn+0x7c/0x1c4 __warn from warn_slowpath_fmt+0x11c/0x1a8 warn_slowpath_fmt from drm_atomic_helper_wait_for_vblanks.part.0+0x2b0/0x2b8 drm_atomic_helper_wait_for_vblanks.part.0 from drm_atomic_helper_commit_tail_rpm+0x7c/0x8c drm_atomic_helper_commit_tail_rpm from commit_tail+0x9c/0x184 commit_tail from drm_atomic_helper_commit+0x168/0x190 drm_atomic_helper_commit from drm_atomic_commit+0xb4/0xe0 drm_atomic_commit from drm_client_modeset_commit_atomic+0x23c/0x27c drm_client_modeset_commit_atomic from drm_client_modeset_commit_locked+0x60/0x1cc drm_client_modeset_commit_locked from drm_client_modeset_commit+0x24/0x40 drm_client_modeset_commit from __drm_fb_helper_restore_fbdev_mode_unlocked+0x9c/0xc4 __drm_fb_helper_restore_fbdev_mode_unlocked from drm_fb_helper_set_par+0x2c/0x3c drm_fb_helper_set_par from fbcon_init+0x3d8/0x550 fbcon_init from visual_init+0xc0/0x108 visual_init from do_bind_con_driver+0x1b8/0x3a4 do_bind_con_driver from do_take_over_console+0x140/0x1ec do_take_over_console from do_fbcon_takeover+0x70/0xd0 do_fbcon_takeover from fbcon_fb_registered+0x19c/0x1ac fbcon_fb_registered from register_framebuffer+0x190/0x21c register_framebuffer from __drm_fb_helper_initial_config_and_unlock+0x350/0x574 __drm_fb_helper_initial_config_and_unlock from exynos_drm_fbdev_client_hotplug+0x6c/0xb0 exynos_drm_fbdev_client_hotplug from drm_client_register+0x58/0x94 drm_client_register from exynos_drm_bind+0x160/0x190 exynos_drm_bind from try_to_bring_up_aggregate_device+0x200/0x2d8 try_to_bring_up_aggregate_device from __component_add+0xb0/0x170 __component_add

from mixer_probe+0x74/0xcc mixer_probe from platform_probe+0x5c/0xb8 platform_probe from really_probe+0xe0/0x3d8 really_probe from __driver_probe_device+0x9c/0x1e4 __driver_probe_device from driver_probe_device+0x30/0xc0 driver_probe_device from __device_attach_driver+0xa8/0x120 __device_attach_driver from bus_for_each_drv+0x80/0xcc bus_for_each_drv from __device_attach+0xac/0x1fc __device_attach from bus_probe_device+0x8c/0x90 bus_probe_device from deferred_probe_work_func+0 ---truncated---

CVE-2024-39493	1 Linux	1 Linux Kernel	2026-05-23	5.5 Medium

In the Linux kernel, the following vulnerability has been resolved: crypto: qat - Fix ADF_DEV_RESET_SYNC memory leak Using completion_done to determine whether the caller has gone away only works after a complete call. Furthermore it's still possible that the caller has not yet called wait_for_completion, resulting in another potential UAF. Fix this by making the caller use cancel_work_sync and then freeing the memory safely.

CVE-2024-35804	1 Linux	1 Linux Kernel	2026-05-23	5.5 Medium

In the Linux kernel, the following vulnerability has been resolved: KVM: x86: Mark target gfn of emulated atomic instruction as dirty When emulating an atomic access on behalf of the guest, mark the target gfn dirty if the CMPXCHG by KVM is attempted and doesn't fault. This fixes a bug where KVM effectively corrupts guest memory during live migration by writing to guest memory without informing userspace that the page is dirty. Marking the page dirty got unintentionally dropped when KVM's emulated CMPXCHG was converted to do a user access. Before that, KVM explicitly mapped the guest page into kernel memory, and marked the page dirty during the unmap phase. Mark the page dirty even if the CMPXCHG fails, as the old data is written back on failure, i.e. the page is still written. The value written is guaranteed to be the same because the operation is atomic, but KVM's ABI is that all writes are dirty logged regardless of the value written. And more importantly, that's what KVM did before the buggy commit. Huge kudos to the folks on the Cc list (and many others), who did all the actual work of triaging and debugging. base-commit: 6769ea8da8a93ed4630f1ce64df6aafcaabfce64

CVE-2024-26858	2 Linux, Redhat	2 Linux Kernel, Enterprise Linux	2026-05-23	4.1 Medium

In the Linux kernel, the following vulnerability has been resolved: net/mlx5e: Use a memory barrier to enforce PTP WQ xmit submission tracking occurs after populating the metadata_map Just simply reordering the functions mlx5e_ptp_metadata_map_put and mlx5e_ptpsq_track_metadata in the mlx5e_txwqe_complete context is not good enough since both the compiler and CPU are free to reorder these two functions. If reordering does occur, the issue that was supposedly fixed by 7e3f3ba97e6c ("net/mlx5e: Track xmit submission to PTP WQ after populating metadata map") will be seen. This will lead to NULL pointer dereferences in mlx5e_ptpsq_mark_ts_cqes_undelivered in the NAPI polling context due to the tracking list being populated before the metadata map.

CVE-2022-49919	1 Linux	1 Linux Kernel	2026-05-23	7 High

In the Linux kernel, the following vulnerability has been resolved: netfilter: nf_tables: release flow rule object from commit path No need to postpone this to the commit release path, since no packets are walking over this object, this is accessed from control plane only. This helped uncovered UAF triggered by races with the netlink notifier.

CVE-2022-49713	1 Linux	1 Linux Kernel	2026-05-23	5.5 Medium

In the Linux kernel, the following vulnerability has been resolved: usb: dwc2: Fix memory leak in dwc2_hcd_init usb_create_hcd will alloc memory for hcd, and we should call usb_put_hcd to free it when platform_get_resource() fails to prevent memory leak. goto error2 label instead error1 to fix this.

CVE-2022-49664	2 Linux, Redhat	2 Linux Kernel, Enterprise Linux	2026-05-23	5.5 Medium

In the Linux kernel, the following vulnerability has been resolved: tipc: move bc link creation back to tipc_node_create Shuang Li reported a NULL pointer dereference crash: [] BUG: kernel NULL pointer dereference, address: 0000000000000068 [] RIP: 0010:tipc_link_is_up+0x5/0x10 [tipc] [] Call Trace: [] <IRQ> [] tipc_bcast_rcv+0xa2/0x190 [tipc] [] tipc_node_bc_rcv+0x8b/0x200 [tipc] [] tipc_rcv+0x3af/0x5b0 [tipc] [] tipc_udp_recv+0xc7/0x1e0 [tipc] It was caused by the 'l' passed into tipc_bcast_rcv() is NULL. When it creates a node in tipc_node_check_dest(), after inserting the new node into hashtable in tipc_node_create(), it creates the bc link. However, there is a gap between this insert and bc link creation, a bc packet may come in and get the node from the hashtable then try to dereference its bc link, which is NULL. This patch is to fix it by moving the bc link creation before inserting into the hashtable. Note that for a preliminary node becoming "real", the bc link creation should also be called before it's rehashed, as we don't create it for preliminary nodes.

CVE-2021-47657	2 Linux, Redhat	2 Linux Kernel, Enterprise Linux	2026-05-23	5.5 Medium

In the Linux kernel, the following vulnerability has been resolved: drm/virtio: Ensure that objs is not NULL in virtio_gpu_array_put_free() If virtio_gpu_object_shmem_init() fails (e.g. due to fault injection, as it happened in the bug report by syzbot), virtio_gpu_array_put_free() could be called with objs equal to NULL. Ensure that objs is not NULL in virtio_gpu_array_put_free(), or otherwise return from the function.

CVE-2026-43494	1 Linux	1 Linux Kernel	2026-05-23	7.0 High

In the Linux kernel, the following vulnerability has been resolved: net/rds: reset op_nents when zerocopy page pin fails When iov_iter_get_pages2() fails in rds_message_zcopy_from_user(), the pinned pages are released with put_page(), and rm->data.op_mmp_znotifier is cleared. But we fail to properly clear rm->data.op_nents. Later when rds_message_purge() is called from rds_sendmsg() the cleanup loop iterates over the incorrectly non zero number of op_nents and frees them again. Fix this by properly resetting op_nents when it should be in rds_message_zcopy_from_user().

CVE-2026-43490	1 Linux	1 Linux Kernel	2026-05-23	8.8 High

In the Linux kernel, the following vulnerability has been resolved: ksmbd: validate inherited ACE SID length smb_inherit_dacl() walks the parent directory DACL loaded from the security descriptor xattr. It verifies that each ACE contains the fixed SID header before using it, but does not verify that the variable-length SID described by sid.num_subauth is fully contained in the ACE. A malformed inheritable ACE can advertise more subauthorities than are present in the ACE. compare_sids() may then read past the ACE. smb_set_ace() also clamps the copied destination SID, but used the unchecked source SID count to compute the inherited ACE size. That could advance the temporary inherited ACE buffer pointer and nt_size accounting past the allocated buffer. Fix this by validating the parent ACE SID count and SID length before using the SID during inheritance. Compute the inherited ACE size from the copied SID so the size matches the bounded destination SID. Reject the inherited DACL if size accumulation would overflow smb_acl.size or the security descriptor allocation size.

CVE-2026-43245	1 Linux	1 Linux Kernel	2026-05-23	7.5 High

In the Linux kernel, the following vulnerability has been resolved: ntfs: ->d_compare() must not block ... so don't use __getname() there. Switch it (and ntfs_d_hash(), while we are at it) to kmalloc(PATH_MAX, GFP_NOWAIT). Yes, ntfs_d_hash() almost certainly can do with smaller allocations, but let ntfs folks deal with that - keep the allocation size as-is for now. Stop abusing names_cachep in ntfs, period - various uses of that thing in there have nothing to do with pathnames; just use k[mz]alloc() and be done with that. For now let's keep sizes as-in, but AFAICS none of the users actually want PATH_MAX.

CVE-2026-43137	1 Linux	1 Linux Kernel	2026-05-23	5.5 Medium

In the Linux kernel, the following vulnerability has been resolved: ASoC: SOF: Intel: hda: Fix NULL pointer dereference If there's a mismatch between the DAI links in the machine driver and the topology, it is possible that the playback/capture widget is not set, especially in the case of loopback capture for echo reference where we use the dummy DAI link. Return the error when the widget is not set to avoid a null pointer dereference like below when the topology is broken. RIP: 0010:hda_dai_get_ops.isra.0+0x14/0xa0 [snd_sof_intel_hda_common]

CVE-2026-31707	1 Linux	1 Linux Kernel	2026-05-23	7.1 High

In the Linux kernel, the following vulnerability has been resolved: ksmbd: validate response sizes in ipc_validate_msg() ipc_validate_msg() computes the expected message size for each response type by adding (or multiplying) attacker-controlled fields from the daemon response to a fixed struct size in unsigned int arithmetic. Three cases can overflow: KSMBD_EVENT_RPC_REQUEST: msg_sz = sizeof(struct ksmbd_rpc_command) + resp->payload_sz; KSMBD_EVENT_SHARE_CONFIG_REQUEST: msg_sz = sizeof(struct ksmbd_share_config_response) + resp->payload_sz; KSMBD_EVENT_LOGIN_REQUEST_EXT: msg_sz = sizeof(struct ksmbd_login_response_ext) + resp->ngroups * sizeof(gid_t); resp->payload_sz is __u32 and resp->ngroups is __s32. Each addition can wrap in unsigned int; the multiplication by sizeof(gid_t) mixes signed and size_t, so a negative ngroups is converted to SIZE_MAX before the multiply. A wrapped value of msg_sz that happens to equal entry->msg_sz bypasses the size check on the next line, and downstream consumers (smb2pdu.c:6742 memcpy using rpc_resp->payload_sz, kmemdup in ksmbd_alloc_user using resp_ext->ngroups) then trust the unverified length. Use check_add_overflow() on the RPC_REQUEST and SHARE_CONFIG_REQUEST paths to detect integer overflow without constraining functional payload size; userspace ksmbd-tools grows NDR responses in 4096-byte chunks for calls like NetShareEnumAll, so a hard transport cap is unworkable on the response side. For LOGIN_REQUEST_EXT, reject resp->ngroups outside the signed [0, NGROUPS_MAX] range up front and report the error from ipc_validate_msg() so it fires at the IPC boundary; with that bound the subsequent multiplication and addition stay well below UINT_MAX. The now-redundant ngroups check and pr_err in ksmbd_alloc_user() are

removed. This is the response-side analogue of aab98e2dbd64 ("ksmbd: fix integer overflows on 32 bit systems"), which hardened the request side.

CVE-2026-31613	1 Linux	1 Linux Kernel	2026-05-23	8.1 High

In the Linux kernel, the following vulnerability has been resolved: smb: client: fix OOB reads parsing symlink error response When a CREATE returns STATUS_STOPPED_ON_SYMLINK, smb2_check_message() returns success without any length validation, leaving the symlink parsers as the only defense against an untrusted server. symlink_data() walks SMB 3.1.1 error contexts with the loop test "p < end", but reads p->ErrorId at offset 4 and p->ErrorDataLength at offset 0. When the server-controlled ErrorDataLength advances p to within 1-7 bytes of end, the next iteration will read past it. When the matching context is found, sym->SymLinkErrorTag is read at offset 4 from p->ErrorContextData with no check that the symlink header itself fits. smb2_parse_symlink_response() then bounds-checks the substitute name using SMB2_SYMLINK_STRUCT_SIZE as the offset of PathBuffer from iov_base. That value is computed as sizeof(smb2_err_rsp) + sizeof(smb2_symlink_err_rsp), which is correct only when ErrorContextCount == 0. With at least one error context the symlink data sits 8 bytes deeper, and each skipped non-matching context shifts it further by 8 + ALIGN(ErrorDataLength, 8). The check is too short, allowing the substitute name read to run past iov_len. The out-of-bound heap bytes are UTF-16-decoded into the symlink target and returned to userspace via readlink(2). Fix this all up by making the loops test require the full context header to fit, rejecting sym if its header runs past end, and bound the substitute name against the actual position of sym->PathBuffer rather than a fixed offset. Because sub_offs and sub_len are 16bits, the pointer math will not overflow here with the new greater-than.