Security Bulletin

Bulletin ID: RITTAL-SA-2026-0018
Project: RITTAL ReOS1 - CMC3
Version: v10.3.0
Compared against: v10.2.1
Generated on: 23.11.2026
Classification: TLP:CLEAR

Summary

This bulletin documents 315 security vulnerabilities (18 Critical, 114 High, 178 Medium, 5 Low) that were fixed in RITTAL ReOS1 - CMC3 version v10.3.0 compared to version v10.2.1. Rittal recommends updating to the latest firmware version.

Table of Contents

1. Affected Products
2. Fixed Vulnerabilities Overview
3. Fixed Vulnerabilities
4. Recommended Actions
5. Contact
6. Disclaimer

Affected Products (Article Numbers)

Article Number	Description
3311.130	Luft/Wasser Wärmetauscher, Rackklimatisierung, T=1000 mm, 30 kW
3311.230	Luft/Wasser Wärmetauscher, Rackklimatisierung, T=1200 mm, 30 kW
3311.260	Luft/Wasser Wärmetauscher, Rackklimatisierung, T=1200 mm, 60 kW
3311.530	Luft/Wasser Wärmetauscher, Reihenklimatisierung, vorgezogen, 30 kW
3311.540	Luft/Wasser Wärmetauscher, Reihenklimatisierung, bündig mit Serverracks, 30 kW
3311.560	Luft/Wasser Wärmetauscher, Reihenklimatisierung, vorgezogen, 60 kW
3312.130	LCP Rack R15 CW, 30 kW, RAL 7035, BHT: 300x2000x1000 mm
3312.230	LCP Rack R15 CW, 30 kW, RAL 7035, BHT: 300x2000x1200 mm
3312.250	LCP Rack R15 CW/Glykol, 35 kW, RAL 7035, BHT: 300x2000x1200 mm
3312.260	LCP Rack R15 CW, 55 kW, RAL 7035, BHT: 300x2000x1200 mm
3312.530	LCP Inline Protruding R15 CW, 30 kW, RAL 7035, BHT: 300x2000x1200 mm
3312.540	LCP Inline Flush R15 CW, 30 kW, RAL 7035, BHT: 300x2000x1200 mm
3312.550	LCP Inline Flush R15 CW/Glykol, 35 kW, RAL 7035, BHT: 300x2000x1200 mm
3312.560	LCP Inline Protruding R15 CW, 55 kW, RAL 7035, BHT: 300x2000x1200 mm
3312.570	LCP Inline Protruding R15 CW/Glykol, 35 kW, RAL 7035, BHT: 300x2000x1200 mm
3313.130	LCP Rack CW, 30 kW, RAL 7035, BHT: 300x2000x1000 mm
3313.230	LCP Rack CW, 30 kW, RAL 7035, BHT: 300x2000x1200 mm
3313.238	SK LCP Rack CW UL, 30kW, RAL 9005, 300x2000x1200
3313.250	LCP Rack CWG, 44 kW, RAL 7035, BHT: 300x2000x1200 mm
3313.260	LCP Rack CW, 53 kW, RAL 7035, BHT: 300x2000x1200 mm
3313.268	SK LCP Rack CW UL, 53kW, 300x2000x1200, RAL 9005
3313.530	LCP Inline CW, 30 kW, RAL 7035, BHT: 300x2000x1200 mm
3313.538	SK LCP Inline CW UL, vorgezogen, 30kW, 300x2000x1200, RAL 9005
3313.548	SK LCP Inline CW UL, bündig, 30kW, 300x2000x1200, RAL 9005
3313.550	LCP Inline CWG, 35 kW, RAL 7035, BHT: 300x2000x1200 mm

Article Number	Description
3313.560	LCP Inline CW, 53 kW, RAL 7035, BHT: 300x2000x1200 mm
3313.568	SK LCP Inline CW UL vorgezogen, 53kW UL, 300x2000x1200, RAL 9005
3313.570	LCP Inline CWG, 44 kW, RAL 7035, BHT: 300x2000x1200 mm
3398.195	Steuermodul für LCP
7030.000	Überwachungssystem CMC III Processing Unit, 32 Sensoren

Fixed Vulnerabilities Overview

All	Critical	High	Medium	Low
315	18	114	178	5

Fixed Vulnerabilities

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-43083	CRITICAL	9.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: ioam6: fix OOB and missing lock When trace->type.bit6 is set: if (trace->type.bit6) { ... queue = skb_get_tx_queue(dev, skb); qdisc = rcu_dereference(queue->qdisc); This code can lead to an out-of-bounds access of the dev->_tx[] array when is_input is true. In such a case, the packet is on the RX path and skb->queue_mapping contains the RX queue index of the ingress device. If the ingress device has more RX queues than the egress device (dev) has TX queues, skb_get_queue_mapping(skb) will exceed dev->num_tx_queues. Add a check to avoid this situation since skb_get_tx_queue() does not clamp the index. This issue has also revealed that per queue visibility cannot be accurate and will be replaced later as a new feature. While at it, add missing lock around qdisc_qstats_qlen_backlog(). The function __ioam6_fill_trace_data() is called from both softirq and process contexts, hence the use of spin_lock_bh() here.
CVE-2026-43198	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: tcp: fix potential race in tcp_v6_syn_recv_sock() Code in tcp_v6_syn_recv_sock() after the call to tcp_v4_syn_recv_sock() is done too late. After tcp_v4_syn_recv_sock(), the child socket is already visible from TCP ehash table and other cpus might use it. Since newinet->pinet6 is still pointing to the listener ipv6_pinfo bad things can happen as syzbot found. Move the problematic code in tcp_v6_mapped_child_init() and call this new helper from tcp_v4_syn_recv_sock() before the ehash insertion. This allows the removal of one tcp_sync_mss(), since tcp_v4_syn_recv_sock() will call it with the correct context.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-46325	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: RDMA/rxe: Fix iova-to-va conversion for MR page sizes != PAGE_SIZE The current implementation incorrectly handles memory regions (MRs) with page sizes different from the system PAGE_SIZE. The core issue is that rxe_set_page() is called with mr->page_size step increments, but the page_list stores individual struct page pointers, each representing PAGE_SIZE of memory. ib_sg_to_page() has ensured that when i>=1 either a) SG[i-1].dma_end and SG[i].dma_addr are contiguous or b) SG[i-1].dma_end and SG[i].dma_addr are mr->page_size aligned. This leads to incorrect iova-to-va conversion in scenarios: 1) page_size < PAGE_SIZE (e.g., MR: 4K, system: 64K): ibmr->iova = 0x181800 sg[0]: dma_addr=0x181800, len=0x800 sg[1]: dma_addr=0x173000, len=0x1000 Access iova = 0x181800 + 0x810 = 0x182010 Expected VA: 0x173010 (second SG, offset 0x10) Before fix: - index = (0x182010 >> 12) - (0x181800 >> 12) = 1 - page_offset = 0x182010 & 0xFFF = 0x10 - xarray[1] stores system page base 0x170000 - Resulting VA: 0x170000 + 0x10 = 0x170010 (wrong) 2) page_size > PAGE_SIZE (e.g., MR: 64K, system: 4K): ibmr->iova = 0x18f800 sg[0]: dma_addr=0x18f800, len=0x800 sg[1]: dma_addr=0x170000, len=0x1000 Access iova = 0x18f800 + 0x810 = 0x190010 Expected VA: 0x170010 (second SG, offset 0x10) Before fix: - index = (0x190010 >> 16) - (0x18f800 >> 16) = 1 - page_offset = 0x190010 & 0xFFFF = 0x10 - xarray[1] stores system page for dma_addr 0x170000 - Resulting VA: system page of 0x170000 + 0x10 = 0x170010 (wrong) Yi Zhang reported a kernel panic[1] years ago related to this defect. Solution: 1. Replace xarray with pre-allocated rxe_mr_page array for sequential indexing (all MR page indices are contiguous) 2. Each rxe_mr_page stores both struct page* and offset within the system page 3. Handle MR page_size != PAGE_SIZE relationships: - page_size > PAGE_SIZE: Split MR pages into multiple system pages - page_size <= PAGE_SIZE: Store offset within system page 4. Add boundary checks and compatibility validation This ensures correct iova-to-va conversion regardless of MR page size and system PAGE_SIZE relationship, while improving performance through array-based sequential access. Tests on 4K and 64K PAGE_SIZE hosts: - rdma-core/pytests $ ./build/bin/run_tests.py --dev eth0_rxe - blktest: $ TIMEOUT=30 QUICK_RUN=1 USE_RXE=1 NVMET_TRTYPES=rdma ./check nvme srp nbd [1] https://lore.kernel.org/all/CAHj4cs9XRqE25jyVw9rj9YugffLn5+f=1znaBEnu1usLOciD+g@mail.gmail.com/T/
CVE-2026-53010	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ksmbd: fix use-after-free in smb2_open during durable reconnect In smb2_open, the call to ksmbd_put_durable_fd(fp) drops the reference to the durable file descriptor early during the durable reconnect process. If an error occurs subsequently (eg, ksmbd_iov_pin_rsp fails) or a scavenger accesses the file, it leads to a use-after-free when accessing fp properties (eg fp->create_time). Move the single put to the end of the function below err_out2 so fp stays valid until smb2_open returns.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-53224	CRITICAL	9.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: sctp: validate embedded INIT chunk and address list lengths in cookie sctp_unpack_cookie() only checked that the embedded INIT chunk length did not exceed the remaining cookie payload, but did not ensure that the INIT chunk is large enough to contain a complete INIT header. A malformed COOKIE_ECHO can therefore carry a truncated INIT chunk whose length field is smaller than sizeof(struct sctp_init_chunk). Later, sctp_process_init() accesses INIT parameters unconditionally, which may lead to out-of-bounds reads. In addition, raw_addr_list_len is not fully validated against the remaining cookie payload. When cookie authentication is disabled, an attacker can supply an oversized raw_addr_list_len and cause sctp_raw_to_bind_addrs() to read beyond the end of the cookie. The address parser also lacks sufficient bounds checks for parameter headers and lengths, allowing malformed address parameters to trigger out-of-bounds reads. Fix this by: - requiring the embedded INIT chunk length to be at least sizeof(struct sctp_init_chunk); - validating that the INIT chunk and raw address list together fit within the cookie payload; - verifying sufficient data exists for each address parameter header and payload before parsing it. Note that sctp_verify_init() must be called after sctp_unpack_cookie() and before sctp_process_init() when cookie authentication is disabled. This will be addressed in a separate patch.
CVE-2026-53246	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: sctp: validate cached peer INIT chunk length in COOKIE_ECHO processing When a listening SCTP server processes a COOKIE_ECHO chunk, the cached peer INIT chunk embedded after the cookie is parsed and its parameters are later walked by sctp_process_init() using sctp_walk_params(). However, the chunk header length of this cached INIT chunk was not validated against the remaining buffer in the COOKIE_ECHO payload. If the length field is inflated, the parameter walk can run beyond the actual received data, leading to out-of-bounds reads and potential memory corruption during later parameter handling (e.g. STATE_COOKIE processing and kmemdup() copies). Add a bounds check in sctp_unpack_cookie() to ensure the cached INIT chunk length does not exceed the available data in the COOKIE_ECHO buffer before it is used.
CVE-2026-10536	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	curl 8.7.1	A use-after-free vulnerability exists in libcurl when an application configures an HTTP/2 stream-dependency tree via `CURLOPT_STREAM_DEPENDS` or `CURLOPT_STREAM_DEPENDS_E`, subsequently invokes `curl_easy_reset()`, and finally terminates the handle with `curl_easy_cleanup()`. During this final cleanup phase, libcurl attempts to access and modify an internal structure that was already freed during the reset operation.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-63825	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: gcov: use atomic counter updates to fix concurrent access crashes GCC's GCOV instrumentation can merge global branch counters with loop induction variables as an optimization. In inflate_fast(), the inner copy loops get transformed so that the GCOV counter value is loaded multiple times to compute the loop base address, start index, and end bound. Since GCOV counters are global (not per-CPU), concurrent execution on different CPUs causes the counter to change between loads, producing inconsistent values and out-of-bounds memory writes. The crash manifests during IPComp (IP Payload Compression) processing when inflate_fast() runs concurrently on multiple CPUs: BUG: unable to handle page fault for address: ffffd0a3c0902ffa RIP: inflate_fast+1431 Call Trace: zlib_inflate __deflate_decompress crypto_comp_decompress ipcomp_decompress [xfrm_ipcomp] ipcomp_input [xfrm_ipcomp] xfrm_input At the crash point, the compiler generated three loads from the same global GCOV counter (__gcov0.inflate_fast+216) to compute base, start, and end for an indexed loop. Another CPU modified the counter between loads, making the values inconsistent - the write went 3.4 MB past a 65 KB buffer. Add -fprofile-update=prefer-atomic to CFLAGS_GCOV at the global level in the top-level Makefile, guarded by a try-run compile test. The test compiles a minimal program with and without -fprofile-update=prefer-atomic using the full KBUILD_CFLAGS, then compares undefined symbols in the resulting object files. If prefer-atomic introduces new undefined references (such as __atomic_fetch_add_8 on i386 or __aarch64_ldadd8_relax on arm64 with outline-atomics), the flag is not added -- the kernel does not link against libatomic. On architectures where GCC inlines 64-bit atomic counter updates (x86_64, s390, ...) the test passes and the flag is enabled, preventing the compiler from merging counters with loop induction variables and fixing the observed concurrent-access crash. On architectures where the flag would introduce libatomic dependencies, it is silently omitted and behaviour is no worse than before this patch. Move the CFLAGS_GCOV block from its original position (before the arch Makefile include) to after the core KBUILD_CFLAGS assignments but before the scripts/Makefile.gcc-plugins include. This placement ensures the try-run test sees arch-specific flags (-m32, -march=, -mno-outline-atomics) while avoiding GCC plugin flags (-fplugin=) that would break the test on clean builds when plugin shared objects do not yet exist.
CVE-2026-63857	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: airoha: Do not read uninitialized fragment address in airoha_dev_xmit() The transmit loop in airoha_dev_xmit() reads fragment address and length during its final iteration, when the loop index equals skb_shinfo(skb)->nr_frags, at which point the fragment data is uninitialized. While these values are never consumed, the read itself is unsafe and may trigger a page fault. Fix this by avoiding the fragment read on the last iteration. Additionally, move the skb pointer from the first to the last used packet descriptor, so that airoha_qdma_tx_napi_poll() defers freeing the skb until the final descriptor is processed.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-63979	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net/handshake: hand off the pinned file reference to accept_doit handshake_req_next() removes the request from the per-net pending list and drops hn_lock before handshake_nl_accept_doit() reads req->hr_sk->sk_socket and dereferences sock->file (once in FD_PREPARE() and again in get_file()). In that window a consumer running tls_handshake_cancel() followed by sockfd_put() (svc_sock_free) or __fput_sync() (xs_reset_transport) releases sock->file. sock_release() then runs sock_orphan(), zeroing sk_socket, and frees the struct socket. The accept-side code either reads NULL through sk_socket or chases freed memory. The submit-side sock_hold() does not prevent this. sk_refcnt protects struct sock, but struct socket and sock->file are independently refcounted via the file descriptor the consumer owns. Pinning sk leaves sock and sock->file unprotected. Retarget the accept-side dereferences at req->hr_file, which was pinned at submit time, instead of req->hr_sk->sk_socket->file. Pinning on its own is not sufficient: a consumer that cancels between handshake_req_next() returning and accept_doit reaching FD_PREPARE() takes the !remove_pending() branch in handshake_req_cancel() and drops hr_file before the accept side takes its own reference. Hand off an additional file reference inside handshake_req_next(), under hn_lock, so the accept side operates on a reference that no concurrent handshake_req_cancel() can revoke. FD_PREPARE() consumes that handed-off reference, either by transferring it to the new fd in fd_publish() or by dropping it in the cleanup destructor on error; the explicit get_file() that previously balanced FD_PREPARE() is therefore redundant and goes away. Update handshake_req_cancel_test2 and _test3 to simulate the FD_PREPARE() consumption with an fput() so the kunit file-count assertions stay balanced.
CVE-2026-64024	CRITICAL	9.4	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: tcp: fix stale per-CPU tcp_tw_isn leak enabling ISN prediction Blamed commit moved the TIME_WAIT-derived ISN from the skb control block to a per-CPU variable, assuming the value would always be consumed by tcp_conn_request() for the same packet that wrote it. That assumption is violated by multiple drop paths between the producer (__this_cpu_write(tcp_tw_isn, isn) in tcp_v{4,6}_rcv()) and the consumer (tcp_conn_request()): - min_ttl / min_hopcount check - xfrm policy check - tcp_inbound_hash() MD5/AO mismatch - tcp_filter() eBPF/SO_ATTACH_FILTER drop - th->syn && th->fin discard in tcp_rcv_state_process() TCP_LISTEN - psp_sk_rx_policy_check() in tcp_v{4,6}_do_rcv() - tcp_checksum_complete() in tcp_v{4,6}_do_rcv() - tcp_v{4,6}_cookie_check() returning NULL When a packet is dropped on any of these paths, tcp_tw_isn is left set. The next SYN processed on the same CPU then consumes the non zero value in tcp_conn_request(), receiving a potentially predictable ISN. This patch moves back tcp_tw_isn to skb->cb[], getting rid of the per-cpu variable. Note that tcp_v{4,6}_fill_cb() do not set it. Very litle impact on overall code size/complexity: $ scripts/bloat-o-meter -t vmlinux.old vmlinux.new add/remove: 0/0 grow/shrink: 2/1 up/down: 8/-15 (-7) Function old new delta tcp_v6_rcv 3038 3042 +4 tcp_v4_rcv 3035 3039 +4 tcp_conn_request 2938 2923 -15 Total: Before=24436060, After=24436053, chg -0.00%

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-64066	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfs: Fix netfs_read_to_pagecache() to pause on subreq failure Fix netfs_read_to_pagecache() so that it pauses the generation of new subrequests if an already-issued subrequest fails.
CVE-2026-64216	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfs: Fix potential UAF in netfs_unlock_abandoned_read_pages() netfs_unlock_abandoned_read_pages(rreq) accesses the index of the folios it is wanting to unlock and compares that to rreq->no_unlock_folio so that it doesn't unlock a folio being read for netfs_perform_write() or netfs_write_begin(). However, given that netfs_unlock_abandoned_read_pages() is called _after_ NETFS_RREQ_IN_PROGRESS is cleared, the one folio that it's not allowed to dereference is the one specified by ->no_unlock_folio as ownership immediately reverts to the caller. Fix this by storing the folio pointer instead and using that rather than the index. Also fix netfs_unlock_read_folio() where the same applies.
CVE-2026-64564	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: sctp: don't free the ASCONF's own transport in DEL-IP processing sctp_process_asconf() caches the transport the ASCONF chunk is processed against in asconf->transport (== chunk->transport, set once in sctp_rcv()). For an ASCONF located through its Address Parameter by __sctp_rcv_asconf_lookup(), that cached transport corresponds to the Address Parameter, which need not be the packet's source address. sctp_process_asconf_param() rejects a DEL-IP for the packet source address (ADDIP D8, SCTP_ERROR_DEL_SRC_IP), but nothing protects asconf->transport. A single ASCONF can therefore carry, in order: [Address Parameter L] [DEL-IP L] [DEL-IP 0.0.0.0] where L differs from the source. The DEL-IP for L passes the D8 check and calls sctp_assoc_rm_peer() on the transport that asconf->transport still points at, freeing it (RCU-deferred). The following wildcard DEL-IP then reuses the now-dangling asconf->transport in sctp_assoc_set_primary() and sctp_assoc_del_nonprimary_peers(): set_primary() dereferences the freed transport (->ipaddr, ->state) and plants the dangling pointer into asoc->peer.primary_path / active_path, and del_nonprimary_peers(), keeping only the pointer that is no longer on the list, removes every real transport, leaving the association with a transport_count of 0 and primary_path/active_path pointing at freed memory. Reject a DEL-IP that targets the transport the ASCONF is being processed against, mirroring the existing source-address guard, so the wildcard branch can never reuse a freed transport.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-63073	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	openssl 3.5.7	Issue summary: OpenSSL CMP response validation passed an unexpected response sender distinguished name directly as the format string to `ERR_raise_data()`. Impact summary: A malicious or intercepted CMP endpoint can crash a CMP client that enforces an expected sender or uses a pinned server certificate whose subject becomes the default expected sender. CWE: CWE-134 (Use of Externally-Controlled Format String) Description: When validating a received CMP message, ossl_cmp_msg_check_update() converts the peer-supplied sender distinguished name with X509_NAME_oneline() and passes it directly as the format argument to ERR_raise_data(). Percent characters survive the conversion, so a sender DN such as "CN=%s%n" reaches BIO_vsnprintf() as an attacker-controlled format string with no matching variadic arguments. This path is only reached when the caller configures an expected sender or pins a server certificate, which is the normal configuration for a CMP client validating server responses. Since the attacker controls the format string but none of the variadic arguments, such specifiers as %s and %n dereference or write through unrelated stack contents and crash the client. The reliable consequence is a denial of service, when the response comes from a malicious or intercepted CMP endpoint. There is no controlled memory write, arbitrary-address read, or reliable path to remote code execution. FIPS impact: no No FIPS modules are affected by this issue, as the CMP protocol implementation is outside the OpenSSL FIPS module boundary.
CVE-2026-75803	CRITICAL	9.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N	openssl 3.5.7	Issue summary: ChaCha20-Poly1305 and AES-OCB decryption with an empty ciphertext can report success without verifying the supplied authentication tag when the operation is finalized by calling the EVP_Cipher() function. Impact summary: Applications calling EVP_Cipher() on an empty ciphertext and expecting the call to check the AEAD tag may accept forged messages. CWE: CWE-354 (Improper Validation of Integrity Check Value) Description: The EVP_Cipher() API call for AEAD ciphers behaves like a one shot encryption and decryption call. It also verifies the AEAD tag after the decryption operation. However for AES-OCB and ChaCha20-Poly1305 ciphers it skipped the AEAD tag verification when an empty ciphertext was passed to the function. The callers of this function might believe that a successful return indicates a valid AEAD tag for these ciphers, even when that has not truly been validated in this case. FIPS impact: no The FIPS modules in 4.0, 3.6, 3.5, 3.4, and 3.0 are not affected by this CVE as the affected algorithms are not FIPS approved and thus not implemented in the FIPS module.
CVE-2026-100075	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: RDMA/srpt: Fix srpt_alloc_rw_ctxs() unwind counters When srpt_alloc_rw_ctxs() fails partway through a multi-buffer indirect descriptor, the unwind path destroys RDMA contexts but leaves stale n_rw_ctx and n_rdma values (and a dangling rw_ctxs pointer). Later sq_wr_avail accounting in srpt_queue_response() or srpt_write_pending() can then subtract the wrong number of send queue credits. Reset the counters and clear rw_ctxs after freeing the heap allocation before returning an error.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98432	CRITICAL	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ipv4: free inet_opt and ireq_opt after an RCU grace period tcp_v4_syn_recv_sock() transfers ownership of ireq->ireq_opt to the child socket (newinet->inet_opt) without copying it. Another cpu can concurrently process a retransmitted SYN for the same request socket, and send a SYNACK from tcp_check_req(). tcp_v4_send_synack() and inet_csk_route_req() read ireq->ireq_opt under rcu_read_lock() only, and ip_build_and_send_pkt() and ip_options_build() then read opt->optlen twice. Note that the SYNACK timer itself is not an issue: it holds a reference on its request socket, and inet_csk_reqsk_queue_drop() calls timer_delete_sync() before the child can be freed. Since commit 079096f103fa ("tcp/dccp: install syn_recv requests into ehash table"), request sockets are processed without holding the listener lock, so nothing prevents the child socket from being freed while the SYNACK is still being built. TCP child sockets do not have SOCK_RCU_FREE, and inet_sock_destruct() frees inet_opt with a plain kfree(), leading to a use-after-free in ip_options_build(). A similar issue exists with request socket migration (net.ipv4.tcp_migrate_req=1, or a BPF_SK_REUSEPORT_SELECT_OR_MIGRATE program). reqsk_timer_handler() clones the request socket with inet_reqsk_clone(), so that the old request socket and its clone share the same ireq_opt, then reqsk_migrate_reset() clears the pointer in the old one. Another cpu holding a reference on the old request socket can still be using these options (sending a SYNACK, or creating a child in tcp_v4_syn_recv_sock()) when the clone is freed, and tcp_v4_reqsk_destructor() also uses a plain kfree(). Readers already use RCU, and other paths replacing inet_opt (do_ip_setsockopt(), cipso_v4_sock_setattr()...) already use kfree_rcu(). Use kfree_rcu() in inet_sock_destruct() and tcp_v4_reqsk_destructor() as well. IPv6 is not affected by the first issue, because tcp_v6_syn_recv_sock() duplicates the options. tcp_v6_reqsk_destructor() has the same migration issue with ipv6_opt, which is only set by CALIPSO. This will be addressed in a separate patch.
CVE-2023-51767	HIGH	7.0	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H	openssh 9.6p1	OpenSSH through 10.0, when common types of DRAM are used, might allow row hammer attacks (for authentication bypass) because the integer value of authenticated in mm_answer_authpassword does not resist flips of a single bit. NOTE: this is applicable to a certain threat model of attacker-victim co-location in which the attacker has user privileges. NOTE: this is disputed by the Supplier, who states "we do not consider it to be the application's responsibility to defend against platform architectural weaknesses."
CVE-2020-25672	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	A memory leak vulnerability was found in Linux kernel in llcp_sock_connect
CVE-2016-2568	HIGH	7.8	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:H	polkit 124	pkexec, when used with --user nonpriv, allows local users to escape to the parent session via a crafted TIOCSTI ioctl call, which pushes characters to the terminal's input buffer.
CVE-2008-3844	HIGH	N/A	N/A	openssh 9.6p1	Certain Red Hat Enterprise Linux (RHEL) 4 and 5 packages for OpenSSH, as signed in August 2008 using a legitimate Red Hat GPG key, contain an externally introduced modification (Trojan Horse) that allows the package authors to have an unknown impact. NOTE: since the malicious packages were not distributed from any official Red Hat sources, the scope of this issue is restricted to users who may have obtained these packages through unofficial distribution points. As of 20080827, no unofficial distributions of this software are known.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-21709	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: kernel: be more careful about dup_mmap() failures and uprobe registering If a memory allocation fails during dup_mmap(), the maple tree can be left in an unsafe state for other iterators besides the exit path. All the locks are dropped before the exit_mmap() call (in mm/mmap.c), but the incomplete mm_struct can be reached through (at least) the rmap finding the vmas which have a pointer back to the mm_struct. Up to this point, there have been no issues with being able to find an mm_struct that was only partially initialised. Syzbot was able to make the incomplete mm_struct fail with recent forking changes, so it has been proven unsafe to use the mm_struct that hasn't been initialised, as referenced in the link below. Although 8ac662f5da19f ("fork: avoid inappropriate uprobe access to invalid mm") fixed the uprobe access, it does not completely remove the race. This patch sets the MMF_OOM_SKIP to avoid the iteration of the vmas on the oom side (even though this is extremely unlikely to be selected as an oom victim in the race window), and sets MMF_UNSTABLE to avoid other potential users from using a partially initialised mm_struct. When registering vmas for uprobe, skip the vmas in an mm that is marked unstable. Modifying a vma in an unstable mm may cause issues if the mm isn't fully initialised.
CVE-2024-52560	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: fs/ntfs3: Mark inode as bad as soon as error detected in mi_enum_attr() Extended the `mi_enum_attr()` function interface with an additional parameter, `struct ntfs_inode *ni`, to allow marking the inode as bad as soon as an error is detected.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-22121	HIGH	8.4	CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ext4: fix out-of-bound read in ext4_xattr_inode_dec_ref_all() There's issue as follows: BUG: KASAN: use-after-free in ext4_xattr_inode_dec_ref_all+0x6ff/0x790 Read of size 4 at addr ffff88807b003000 by task syz-executor.0/15172 CPU: 3 PID: 15172 Comm: syz-executor.0 Call Trace: __dump_stack lib/dump_stack.c:82 [inline] dump_stack+0xbe/0xfd lib/dump_stack.c:123 print_address_description.constprop.0+0x1e/0x280 mm/kasan/report.c:400 __kasan_report.cold+0x6c/0x84 mm/kasan/report.c:560 kasan_report+0x3a/0x50 mm/kasan/report.c:585 ext4_xattr_inode_dec_ref_all+0x6ff/0x790 fs/ext4/xattr.c:1137 ext4_xattr_delete_inode+0x4c7/0xda0 fs/ext4/xattr.c:2896 ext4_evict_inode+0xb3b/0x1670 fs/ext4/inode.c:323 evict+0x39f/0x880 fs/inode.c:622 iput_final fs/inode.c:1746 [inline] iput fs/inode.c:1772 [inline] iput+0x525/0x6c0 fs/inode.c:1758 ext4_orphan_cleanup fs/ext4/super.c:3298 [inline] ext4_fill_super+0x8c57/0xba40 fs/ext4/super.c:5300 mount_bdev+0x355/0x410 fs/super.c:1446 legacy_get_tree+0xfe/0x220 fs/fs_context.c:611 vfs_get_tree+0x8d/0x2f0 fs/super.c:1576 do_new_mount fs/namespace.c:2983 [inline] path_mount+0x119a/0x1ad0 fs/namespace.c:3316 do_mount+0xfc/0x110 fs/namespace.c:3329 __do_sys_mount fs/namespace.c:3540 [inline] __se_sys_mount+0x219/0x2e0 fs/namespace.c:3514 do_syscall_64+0x33/0x40 arch/x86/entry/common.c:46 entry_SYSCALL_64_after_hwframe+0x67/0xd1 Memory state around the buggy address: ffff88807b002f00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ffff88807b002f80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >ffff88807b003000: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ^ ffff88807b003080: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ffff88807b003100: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff Above issue happens as ext4_xattr_delete_inode() isn't check xattr is valid if xattr is in inode. To solve above issue call xattr_check_inode() check if xattr if valid in inode. In fact, we can directly verify in ext4_iget_extra_inode(), so that there is no divergent verification.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-22109	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ax25: Remove broken autobind Binding AX25 socket by using the autobind feature leads to memory leaks in ax25_connect() and also refcount leaks in ax25_release(). Memory leak was detected with kmemleak: ========== unreferenced object 0xffff8880253cd680 (size 96): backtrace: __kmalloc_node_track_caller_noprof (./include/linux/kmemleak.h:43) kmemdup_noprof (mm/util.c:136) ax25_rt_autobind (net/ax25/ax25_route.c:428) ax25_connect (net/ax25/af_ax25.c:1282) __sys_connect_file (net/socket.c:2045) __sys_connect (net/socket.c:2064) __x64_sys_connect (net/socket.c:2067) do_syscall_64 (arch/x86/entry/common.c:52 arch/x86/entry/common.c:83) entry_SYSCALL_64_after_hwframe (arch/x86/entry/entry_64.S:130) ========== When socket is bound, refcounts must be incremented the way it is done in ax25_bind() and ax25_setsockopt() (SO_BINDTODEVICE). In case of autobind, the refcounts are not incremented. This bug leads to the following issue reported by Syzkaller: ========== ax25_connect(): syz-executor318 uses autobind, please contact jreuter@yaina.de ----------[cut here]---------- refcount_t: decrement hit 0; leaking memory. WARNING: CPU: 0 PID: 5317 at lib/refcount.c:31 refcount_warn_saturate+0xfa/0x1d0 lib/refcount.c:31 Modules linked in: CPU: 0 UID: 0 PID: 5317 Comm: syz-executor318 Not tainted 6.14.0-rc4-syzkaller-00278-gece144f151ac #0 Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.16.3-debian-1.16.3-2~bpo12+1 04/01/2014 RIP: 0010:refcount_warn_saturate+0xfa/0x1d0 lib/refcount.c:31 ... Call Trace: __refcount_dec include/linux/refcount.h:336 [inline] refcount_dec include/linux/refcount.h:351 [inline] ref_tracker_free+0x6af/0x7e0 lib/ref_tracker.c:236 netdev_tracker_free include/linux/netdevice.h:4302 [inline] netdev_put include/linux/netdevice.h:4319 [inline] ax25_release+0x368/0x960 net/ax25/af_ax25.c:1080 __sock_release net/socket.c:647 [inline] sock_close+0xbc/0x240 net/socket.c:1398 __fput+0x3e9/0x9f0 fs/file_table.c:464 __do_sys_close fs/open.c:1580 [inline] __se_sys_close fs/open.c:1565 [inline] __x64_sys_close+0x7f/0x110 fs/open.c:1565 do_syscall_x64 arch/x86/entry/common.c:52 [inline] do_syscall_64+0xf3/0x230 arch/x86/entry/common.c:83 entry_SYSCALL_64_after_hwframe+0x77/0x7f ... ========== Considering the issues above and the comments left in the code that say: "check if we can remove this feature. It is broken."; "autobinding in this may or may not work"; - it is better to completely remove this feature than to fix it because it is broken and leads to various kinds of memory bugs. Now calling connect() without first binding socket will result in an error (-EINVAL). Userspace software that relies on the autobind feature might get broken. However, this feature does not seem widely used with this specific driver as it was not reliable at any point of time, and it is already broken anyway. E.g. ax25-tools and ax25-apps packages for popular distributions do not use the autobind feature for AF_AX25. Found by Linux Verification Center (linuxtesting.org) with Syzkaller.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-38137	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: PCI/pwrctrl: Cancel outstanding rescan work when unregistering It's possible to trigger use-after-free here by: (a) forcing rescan_work_func() to take a long time and (b) utilizing a pwrctrl driver that may be unloaded for some reason Cancel outstanding work to ensure it is finished before we allow our data structures to be cleaned up. [bhelgaas: tidy commit log]

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-38248	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bridge: mcast: Fix use-after-free during router port configuration The bridge maintains a global list of ports behind which a multicast router resides. The list is consulted during forwarding to ensure multicast packets are forwarded to these ports even if the ports are not member in the matching MDB entry. When per-VLAN multicast snooping is enabled, the per-port multicast context is disabled on each port and the port is removed from the global router port list: # ip link add name br1 up type bridge vlan_filtering 1 mcast_snooping 1 # ip link add name dummy1 up master br1 type dummy # ip link set dev dummy1 type bridge_slave mcast_router 2 $ bridge -d mdb show | grep router router ports on br1: dummy1 # ip link set dev br1 type bridge mcast_vlan_snooping 1 $ bridge -d mdb show | grep router However, the port can be re-added to the global list even when per-VLAN multicast snooping is enabled: # ip link set dev dummy1 type bridge_slave mcast_router 0 # ip link set dev dummy1 type bridge_slave mcast_router 2 $ bridge -d mdb show | grep router router ports on br1: dummy1 Since commit 4b30ae9adb04 ("net: bridge: mcast: re-implement br_multicast_{enable, disable}_port functions"), when per-VLAN multicast snooping is enabled, multicast disablement on a port will disable the per-{port, VLAN} multicast contexts and not the per-port one. As a result, a port will remain in the global router port list even after it is deleted. This will lead to a use-after-free [1] when the list is traversed (when adding a new port to the list, for example): # ip link del dev dummy1 # ip link add name dummy2 up master br1 type dummy # ip link set dev dummy2 type bridge_slave mcast_router 2 Similarly, stale entries can also be found in the per-VLAN router port list. When per-VLAN multicast snooping is disabled, the per-{port, VLAN} contexts are disabled on each port and the port is removed from the per-VLAN router port list: # ip link add name br1 up type bridge vlan_filtering 1 mcast_snooping 1 mcast_vlan_snooping 1 # ip link add name dummy1 up master br1 type dummy # bridge vlan add vid 2 dev dummy1 # bridge vlan global set vid 2 dev br1 mcast_snooping 1 # bridge vlan set vid 2 dev dummy1 mcast_router 2 $ bridge vlan global show dev br1 vid 2 | grep router router ports: dummy1 # ip link set dev br1 type bridge mcast_vlan_snooping 0 $ bridge vlan global show dev br1 vid 2 | grep router However, the port can be re-added to the per-VLAN list even when per-VLAN multicast snooping is disabled: # bridge vlan set vid 2 dev dummy1 mcast_router 0 # bridge vlan set vid 2 dev dummy1 mcast_router 2 $ bridge vlan global show dev br1 vid 2 | grep router router ports: dummy1 When the VLAN is deleted from the port, the per-{port, VLAN} multicast context will not be disabled since multicast snooping is not enabled on the VLAN. As a result, the port will remain in the per-VLAN router port list even after it is no longer member in the VLAN. This will lead to a use-after-free [2] when the list is traversed (when adding a new port to the list, for example): # ip link add name dummy2 up master br1 type dummy # bridge vlan add vid 2 dev dummy2 # bridge vlan del vid 2 dev dummy1 # bridge vlan set vid 2 dev dummy2 mcast_router 2 Fix these issues by removing the port from the relevant (global or per-VLAN) router port list in br_multicast_port_ctx_deinit(). The function is invoked during port deletion with the per-port multicast context and during VLAN deletion with the per-{port, VLAN} multicast context. Note that deleting the multicast router timer is not enough as it only takes care of the temporary multicast router states (1 or 3) and not the permanent one (2). [1] BUG: KASAN: slab-out-of-bounds in br_multicast_add_router.part.0+0x3f1/0x560 Write of size 8 at addr ffff888004a67328 by task

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-38584	HIGH	7.0	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: padata: Fix pd UAF once and for all There is a race condition/UAF in padata_reorder that goes back to the initial commit. A reference count is taken at the start of the process in padata_do_parallel, and released at the end in padata_serial_worker. This reference count is (and only is) required for padata_replace to function correctly. If padata_replace is never called then there is no issue. In the function padata_reorder which serves as the core of padata, as soon as padata is added to queue->serial.list, and the associated spin lock released, that padata may be processed and the reference count on pd would go away. Fix this by getting the next padata before the squeue->serial lock is released. In order to make this possible, simplify padata_reorder by only calling it once the next padata arrives.
CVE-2025-39677	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net/sched: Fix backlog accounting in qdisc_dequeue_internal This issue applies for the following qdiscs: hhf, fq, fq_codel, and fq_pie, and occurs in their change handlers when adjusting to the new limit. The problem is the following in the values passed to the subsequent qdisc_tree_reduce_backlog call given a tbf parent: When the tbf parent runs out of tokens, skbs of these qdiscs will be placed in gso_skb. Their peek handlers are qdisc_peek_dequeued, which accounts for both qlen and backlog. However, in the case of qdisc_dequeue_internal, ONLY qlen is accounted for when pulling from gso_skb. This means that these qdiscs are missing a qdisc_qstats_backlog_dec when dropping packets to satisfy the new limit in their change handlers. One can observe this issue with the following (with tc patched to support a limit of 0): export TARGET=fq tc qdisc del dev lo root tc qdisc add dev lo root handle 1: tbf rate 8bit burst 100b latency 1ms tc qdisc replace dev lo handle 3: parent 1:1 $TARGET limit 1000 echo "; echo 'add child'; tc -s -d qdisc show dev lo ping -I lo -f -c2 -s32 -W0.001 127.0.0.1 2>&1 >/dev/null echo "; echo 'after ping'; tc -s -d qdisc show dev lo tc qdisc change dev lo handle 3: parent 1:1 $TARGET limit 0 echo "; echo 'after limit drop'; tc -s -d qdisc show dev lo tc qdisc replace dev lo handle 2: parent 1:1 sfq echo "; echo 'post graft'; tc -s -d qdisc show dev lo The second to last show command shows 0 packets but a positive number (74) of backlog bytes. The problem becomes clearer in the last show command, where qdisc_purge_queue triggers qdisc_tree_reduce_backlog with the positive backlog and causes an underflow in the tbf parent's backlog (4096 Mb instead of 0). To fix this issue, the codepath for all clients of qdisc_dequeue_internal has been simplified: codel, pie, hhf, fq, fq_pie, and fq_codel. qdisc_dequeue_internal handles the backlog adjustments for all cases that do not directly use the dequeue handler. The old fq_codel_change limit adjustment loop accumulated the arguments to the subsequent qdisc_tree_reduce_backlog call through the cstats field. However, this is confusing and error prone as fq_codel_dequeue could also potentially mutate this field (which qdisc_dequeue_internal calls in the non gso_skb case), so we have unified the code here with other qdiscs.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-39905	HIGH	7.0	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: phylink: add lock for serializing concurrent pl->phydev writes with resolver Currently phylink_resolve() protects itself against concurrent phylink_bringup_phy() or phylink_disconnect_phy() calls which modify pl->phydev by relying on pl->state_mutex. The problem is that in phylink_resolve(), pl->state_mutex is in a lock inversion state with pl->phydev->lock. So pl->phydev->lock needs to be acquired prior to pl->state_mutex. But that requires dereferencing pl->phydev in the first place, and without pl->state_mutex, that is racy. Hence the reason for the extra lock. Currently it is redundant, but it will serve a functional purpose once mutex_lock(&phy->lock) will be moved outside of the mutex_lock(&pl->state_mutex) section. Another alternative considered would have been to let phylink_resolve() acquire the rtnl_mutex, which is also held when phylink_bringup_phy() and phylink_disconnect_phy() are called. But since phylink_disconnect_phy() runs under rtnl_lock(), it would deadlock with phylink_resolve() when calling flush_work(&pl->resolve). Additionally, it would have been undesirable because it would have unnecessarily blocked many other call paths as well in the entire kernel, so the smaller-scoped lock was preferred.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-71152	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: dsa: properly keep track of conduit reference Problem description ---------- DSA has a mumbo-jumbo of reference handling of the conduit net device and its kobject which, sadly, is just wrong and doesn't make sense. There are two distinct problems. 1. The OF path, which uses of_find_net_device_by_node(), never releases the elevated refcount on the conduit's kobject. Nominally, the OF and non-OF paths should result in objects having identical reference counts taken, and it is already suspicious that dsa_dev_to_net_device() has a put_device() call which is missing in dsa_port_parse_of(), but we can actually even verify that an issue exists. With CONFIG_DEBUG_KOBJECT_RELEASE=y, if we run this command "before" and "after" applying this patch: (unbind the conduit driver for net device eno2) echo 0000:00:00.2 > /sys/bus/pci/drivers/fsl_enetc/unbind we see these lines in the output diff which appear only with the patch applied: kobject: 'eno2' (ffff002009a3a6b8): kobject_release, parent 0000000000000000 (delayed 1000) kobject: '109' (ffff0020099d59a0): kobject_release, parent 0000000000000000 (delayed 1000) 2. After we find the conduit interface one way (OF) or another (non-OF), it can get unregistered at any time, and DSA remains with a long-lived, but in this case stale, cpu_dp->conduit pointer. Holding the net device's underlying kobject isn't actually of much help, it just prevents it from being freed (but we never need that kobject directly). What helps us to prevent the net device from being unregistered is the parallel netdev reference mechanism (dev_hold() and dev_put()). Actually we actually use that netdev tracker mechanism implicitly on user ports since commit 2f1e8ea726e9 ("net: dsa: link interfaces with the DSA master to get rid of lockdep warnings"), via netdev_upper_dev_link(). But time still passes at DSA switch probe time between the initial of_find_net_device_by_node() code and the user port creation time, time during which the conduit could unregister itself and DSA wouldn't know about it. So we have to run of_find_net_device_by_node() under rtnl_lock() to prevent that from happening, and release the lock only with the netdev tracker having acquired the reference. Do we need to keep the reference until dsa_unregister_switch() / dsa_switch_shutdown()? 1: Maybe yes. A switch device will still be registered even if all user ports failed to probe, see commit 86f8b1c01a0a ("net: dsa: Do not make user port errors fatal"), and the cpu_dp->conduit pointers remain valid. I haven't audited all call paths to see whether they will actually use the conduit in lack of any user port, but if they do, it seems safer to not rely on user ports for that reference. 2. Definitely yes. We support changing the conduit which a user port is associated to, and we can get into a situation where we've moved all user ports away from a conduit, thus no longer hold any reference to it via the net device tracker. But we shouldn't let it go nonetheless - see the next change in relation to dsa_tree_find_first_conduit() and LAG conduits which disappear. We have to be prepared to return to the physical conduit, so the CPU port must explicitly keep another reference to it. This is also to say: the user ports and their CPU ports may not always keep a reference to the same conduit net device, and both are needed. As for the conduit's kobject for the /sys/class/net/ entry, we don't care about it, we can release it as soon as we hold the net device object itself. History and blame attribution ---------- The code has been refactored so many times, it is very difficult to follow and properly attribute a blame, but I'll try to make a short history which I hope to be correct. We have two distinct probing paths: - one for OF, introduced in 2016 i --- truncated---

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-23066	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: rxrpc: Fix recvmsg() unconditional requeue If rxrpc_recvmsg() fails because MSG_DONTWAIT was specified but the call at the front of the recvmsg queue already has its mutex locked, it requeues the call - whether or not the call is already queued. The call may be on the queue because MSG_PEEK was also passed and so the call was not dequeued or because the I/O thread requeued it. The unconditional requeue may then corrupt the recvmsg queue, leading to things like UAFs or refcount underruns. Fix this by only requeuing the call if it isn't already on the queue - and moving it to the front if it is already queued. If we don't queue it, we have to put the ref we obtained by dequeuing it. Also, MSG_PEEK doesn't dequeue the call so shouldn't call rxrpc_notify_socket() for the call if we didn't use up all the data on the queue, so fix that also.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-23102	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: arm64/fpsimd: signal: Fix restoration of SVE context When SME is supported, Restoring SVE signal context can go wrong in a few ways, including placing the task into an invalid state where the kernel may read from out-of-bounds memory (and may potentially take a fatal fault) and/or may kill the task with a SIGKILL. (1) Restoring a context with SVE_SIG_FLAG_SM set can place the task into an invalid state where SVCR.SM is set (and sve_state is non-NULL) but TIF_SME is clear, consequently resuting in out-of-bounds memory reads and/or killing the task with SIGKILL. This can only occur in unusual (but legitimate) cases where the SVE signal context has either been modified by userspace or was saved in the context of another task (e.g. as with CRIU), as otherwise the presence of an SVE signal context with SVE_SIG_FLAG_SM implies that TIF_SME is already set. While in this state, task_fpsimd_load() will NOT configure SMCR_ELx (leaving some arbitrary value configured in hardware) before restoring SVCR and attempting to restore the streaming mode SVE registers from memory via sve_load_state(). As the value of SMCR_ELx.LEN may be larger than the task's streaming SVE vector length, this may read memory outside of the task's allocated sve_state, reading unrelated data and/or triggering a fault. While this can result in secrets being loaded into streaming SVE registers, these values are never exposed. As TIF_SME is clear, fpsimd_bind_task_to_cpu() will configure CPACR_ELx.SMEN to trap EL0 accesses to streaming mode SVE registers, so these cannot be accessed directly at EL0. As fpsimd_save_user_state() verifies the live vector length before saving (S)SVE state to memory, no secret values can be saved back to memory (and hence cannot be observed via ptrace, signals, etc). When the live vector length doesn't match the expected vector length for the task, fpsimd_save_user_state() will send a fatal SIGKILL signal to the task. Hence the task may be killed after executing userspace for some period of time. (2) Restoring a context with SVE_SIG_FLAG_SM clear does not clear the task's SVCR.SM. If SVCR.SM was set prior to restoring the context, then the task will be left in streaming mode unexpectedly, and some register state will be combined inconsistently, though the task will be left in legitimate state from the kernel's PoV. This can only occur in unusual (but legitimate) cases where ptrace has been used to set SVCR.SM after entry to the sigreturn syscall, as syscall entry clears SVCR.SM. In these cases, the the provided SVE register data will be loaded into the task's sve_state using the non-streaming SVE vector length and the FPSIMD registers will be merged into this using the streaming SVE vector length. Fix (1) by setting TIF_SME when setting SVCR.SM. This also requires ensuring that the task's sme_state has been allocated, but as this could contain live ZA state, it should not be zeroed. Fix (2) by clearing SVCR.SM when restoring a SVE signal context with SVE_SIG_FLAG_SM clear. For consistency, I've pulled the manipulation of SVCR, TIF_SVE, TIF_SME, and fp_type earlier, immediately after the allocation of sve_state/sme_state, before the restore of the actual register state. This makes it easier to ensure that these are always modified consistently, even if a fault is taken while reading the register data from the signal context. I do not expect any software to depend on the exact state restored when a fault is taken while reading the context.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-23272	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: nf_tables: unconditionally bump set->nelems before insertion In case that the set is full, a new element gets published then removed without waiting for the RCU grace period, while RCU reader can be walking over it already. To address this issue, add the element transaction even if set is full, but toggle the set_full flag to report -ENFILE so the abort path safely unwinds the set to its previous state. As for element updates, decrement set->nelems to restore it. A simpler fix is to call synchronize_rcu() in the error path. However, with a large batch adding elements to already maxed-out set, this could cause noticeable slowdown of such batches.
CVE-2026-23444	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: mac80211: always free skb on ieee80211_tx_prepare_skb() failure ieee80211_tx_prepare_skb() has three error paths, but only two of them free the skb. The first error path (ieee80211_tx_prepare() returning TX_DROP) does not free it, while invoke_tx_handlers() failure and the fragmentation check both do. Add kfree_skb() to the first error path so all three are consistent, and remove the now-redundant frees in callers (ath9k, mt76, mac80211_hwsim) to avoid double-free. Document the skb ownership guarantee in the function's kdoc.
CVE-2026-31663	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: xfrm: hold dev ref until after transport_finish NF_HOOK After async crypto completes, xfrm_input_resume() calls dev_put() immediately on re-entry before the skb reaches transport_finish. The skb->dev pointer is then used inside NF_HOOK and its okfn, which can race with device teardown. Remove the dev_put from the async resumption entry and instead drop the reference after the NF_HOOK call in transport_finish, using a saved device pointer since NF_HOOK may consume the skb. This covers NF_DROP, NF_QUEUE and NF_STOLEN paths that skip the okfn. For non-transport exits (decaps, gro, drop) and secondary async return points, release the reference inline when async is set.
CVE-2026-5435	HIGH	7.3	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L	glibc 2.39+git	The deprecated functions ns_printtrf, ns_printtr and fp_nquery in the GNU C Library version 2.2 and newer fail to enforce the caller-supplied buffer length, and can result in an out-of-bounds write when printing TSIG records.
CVE-2026-43048	HIGH	8.8	CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: HID: core: Mitigate potential OOB by removing bogus memset() The memset() in hid_report_raw_event() has the good intention of clearing out bogus data by zeroing the area from the end of the incoming data string to the assumed end of the buffer. However, as we have previously seen, doing so can easily result in OOB reads and writes in the subsequent thread of execution. The current suggestion from one of the HID maintainers is to remove the memset() and simply return if the incoming event buffer size is not large enough to fill the associated report. Suggested-by Benjamin Tissoires [bentiss: changed the return value]
CVE-2026-43101	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ipv6: ioam: fix potential NULL dereferences in __ioam6_fill_trace_data() We need to check __in6_dev_get() for possible NULL value, as suggested by Yiming Qian. Also add skb_dst_dev_rcu() instead of skb_dst_dev(), and two missing READ_ONCE(). Note that @dev can't be NULL.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-43116	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: ctnetlink: ensure safe access to master conntrack Holding reference on the expectation is not sufficient, the master conntrack object can just go away, making exp->master invalid. To access exp->master safely: - Grab the nf_conntrack_expect_lock, this gets serialized with clean_from_lists() which also holds this lock when the master conntrack goes away. - Hold reference on master conntrack via nf_conntrack_find_get(). Not so easy since the master tuple to look up for the master conntrack is not available in the existing problematic paths. This patch goes for extending the nf_conntrack_expect_lock section to address this issue for simplicity, in the cases that are described below this is just slightly extending the lock section. The add expectation command already holds a reference to the master conntrack from ctnetlink_create_expect(). However, the delete expectation command needs to grab the spinlock before looking up for the expectation. Expand the existing spinlock section to address this to cover the expectation lookup. Note that, the nf_ct_expect_iterate_net() calls already grabs the spinlock while iterating over the expectation table, which is correct. The get expectation command needs to grab the spinlock to ensure master conntrack does not go away. This also expands the existing spinlock section to cover the expectation lookup too. I needed to move the netlink skb allocation out of the spinlock to keep it GFP_KERNEL. For the expectation events, the IPEXP_DESTROY event is already delivered under the spinlock, just move the delivery of IPEXP_NEW under the spinlock too because the master conntrack event cache is reached through exp->master. While at it, add lockdep notations to help identify what codepaths need to grab the spinlock.
CVE-2026-43245	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	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-2026-43263	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: media: chips-media: wave5: Fix Null reference while testing fluster When multi instances are created/destroyed, many interrupts happens and structures for decoder are removed. "struct vpu_instance" this structure is shared for all flow in the decoder, so if the structure is not protected by lock, Null dereference could happens sometimes. IRQ Handler was spilt to two phases and Lock was added as well.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-43303	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: mm/page_alloc: clear page->private in free_pages_prepare() Several subsystems (slub, shmem, ttm, etc.) use page->private but don't clear it before freeing pages. When these pages are later allocated as high-order pages and split via split_page(), tail pages retain stale page->private values. This causes a use-after-free in the swap subsystem. The swap code uses page->private to track swap count continuations, assuming freshly allocated pages have page->private == 0. When stale values are present, swap_count_continued() incorrectly assumes the continuation list is valid and iterates over uninitialized page->lru containing LIST_POISON values, causing a crash: KASAN: maybe wild-memory-access in range [0xdead000000000100-0xdead000000000107] RIP: 0010:__do_sys_swapoff+0x1151/0x1860 Fix this by clearing page->private in free_pages_prepare(), ensuring all freed pages have clean state regardless of previous use.
CVE-2026-5773	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	curl 8.7.1	libcurl might in some circumstances reuse the wrong connection for SMB(S) transfers. libcurl features a pool of recent connections so that subsequent requests can reuse an existing connection to avoid overhead. When reusing a connection a range of criteria must be met. Due to a logical error in the code, a network transfer operation that was requested by an application could wrongfully reuse an existing SMB connection to the same server that was using a different "share" than the new subsequent transfer should. This could in unlucky situations lead to the download of the wrong file or the upload of a file to the wrong place. When this happens, the same credentials are used and the server name is the same.
CVE-2026-6276	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	curl 8.7.1	Using libcurl, when a custom `Host:` header is first set for an HTTP request and a second request is subsequently done using the same *easy handle* but without the custom `Host:` header set, the second request would use stale information and pass on cookies meant for the first host in the second request. Leak them.
CVE-2026-45932	HIGH	7.3	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: Fix tcx/netkit detach permissions when prog fd isn't given This commit fixes a security issue where BPF_PROG_DETACH on tcx or netkit devices could be executed by any user when no program fd was provided, bypassing permission checks. The fix adds a capability check for CAP_NET_ADMIN or CAP_SYS_ADMIN in this case.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-45944	HIGH	7.5	CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: iommu/vt-d: Clear Present bit before tearing down context entry When tearing down a context entry, the current implementation zeros the entire 128-bit entry using multiple 64-bit writes. This creates a window where the hardware can fetch a "torn" entry — where some fields are already zeroed while the 'Present' bit is still set — leading to unpredictable behavior or spurious faults. While x86 provides strong write ordering, the compiler may reorder writes to the two 64-bit halves of the context entry. Even without compiler reordering, the hardware fetch is not guaranteed to be atomic with respect to multiple CPU writes. Align with the "Guidance to Software for Invalidations" in the VT-d spec (Section 6.5.3.3) by implementing the recommended ownership handshake: 1. Clear only the 'Present' (P) bit of the context entry first to signal the transition of ownership from hardware to software. 2. Use dma_wmb() to ensure the cleared bit is visible to the IOMMU. 3. Perform the required cache and context-cache invalidation to ensure hardware no longer has cached references to the entry. 4. Fully zero out the entry only after the invalidation is complete. Also, add a dma_wmb() to context_set_present() to ensure the entry is fully initialized before the 'Present' bit becomes visible.
CVE-2026-45956	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/exynos: vidi: use priv->vidi_dev for ctx lookup in vidi_connection_ioctl() vidi_connection_ioctl() retrieves the driver_data from drm_dev->dev to obtain a struct vidi_context pointer. However, drm_dev->dev is the exynos-drm master device, and the driver_data contained therein is not the vidi component device, but a completely different device. This can lead to various bugs, ranging from null pointer dereferences and garbage value accesses to, in unlucky cases, out-of-bounds errors, use-after-free errors, and more. To resolve this issue, we need to store/delete the vidi device pointer in exynos_drm_private->vidi_dev during bind/unbind, and then read this exynos_drm_private->vidi_dev within ioctl() to obtain the correct struct vidi_context pointer.
CVE-2026-46093	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: mm/vmalloc: take vmap_purge_lock in shrinker decay_va_pool_node() can be invoked concurrently from two paths: __purge_vmap_area_lazy() when pools are being purged, and the shrinker via vmap_node_shrink_scan(). However, decay_va_pool_node() is not safe to run concurrently, and the shrinker path currently lacks serialization, leading to races and possible leaks. Protect decay_va_pool_node() by taking vmap_purge_lock in the shrinker path to ensure serialization with purge users.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-46175	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: f2fs: fix fsck inconsistency caused by FGGC of node block During FGGC node block migration, fsck may incorrectly treat the migrated node block as fsync-written data. The reproduction scenario: root@vm:/mnt/f2fs# seq 1 2048 | xargs -n 1 ./test_sync // write inline inode and sync root@vm:/mnt/f2fs# rm -f 1 root@vm:/mnt/f2fs# sync root@vm:/mnt/f2fs# f2fs_io gc_range // move data block in sync mode and not write CP SPO, "fsck --dry-run" find inode has already checkpointed but still with DENT_BIT_SHIFT set The root cause is that GC does not clear the dentry mark and fsync mark during node block migration, leading fsck to misinterpret them as user-issued fsync writes. In BGGC mode, node block migration is handled by f2fs_sync_node_pages(), which guarantees the dentry and fsync marks are cleared before writing. This patch move the set/clear of the fsync|dentry marks into __write_node_folio to make the logic clearer, and ensures the fsync|dentry mark is cleared in FGGC.
CVE-2026-46181	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: RDMA/mlx4: Fix mis-use of RCU in mlx4_srq_event() Sashiko points out the radix_tree itself is RCU safe, but nothing ever frees the mlx4_srq struct with RCU, and it isn't even accessed within the RCU critical section. It also will crash if an event is delivered before the srq object is finished initializing. Use the spinlock since it isn't easy to make RCU work, use refcount_inc_not_zero() to protect against partially initialized objects, and order the refcount_set() to be after the srq is fully initialized.
CVE-2026-46242	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: eventpoll: fix ep_remove struct eventpoll / struct file UAF ep_remove() (via ep_remove_file()) cleared file->f_ep under file->f_lock but then kept using @file inside the critical section (is_file_epoll(), hlist_del_rcu() through the head, spin_unlock). A concurrent __fput() taking the eventpoll_release() fastpath in that window observed the transient NULL, skipped eventpoll_release_file() and ran to f_op->release / file_free(). For the epoll-watches-epoll case, f_op->release is ep_eventpoll_release() -> ep_clear_and_put() -> ep_free(), which kfree()s the watched struct eventpoll. Its embedded ->refs hlist_head is exactly where epi->fllink.pprev points, so the subsequent hlist_del_rcu()'s "*pprev = next" scribbles into freed kmalloc-192 memory. In addition, struct file is SLAB_TYPESAFE_BY_RCU, so the slot backing @file could be recycled by alloc_empty_file() -- reinitializing f_lock and f_ep -- while ep_remove() is still nominally inside that lock. The upshot is an attacker-controllable kmem_cache_free() against the wrong slab cache. Pin @file via epi_fget() at the top of ep_remove() and gate the critical section on the pin succeeding. With the pin held @file cannot reach refcount zero, which holds __fput() off and transitively keeps the watched struct eventpoll alive across the hlist_del_rcu() and the f_lock use, closing both UAFs. If the pin fails @file has already reached refcount zero and its __fput() is in flight. Because we bailed before clearing f_ep, that path takes the eventpoll_release() slow path into eventpoll_release_file() and blocks on ep->mtx until the waiter side's ep_clear_and_put() drops it. The bailed epi's share of ep->refcount stays intact, so the trailing ep_refcount_dec_and_test() in ep_clear_and_put() cannot free the eventpoll out from under eventpoll_release_file(); the orphaned epi is then cleaned up there. A successful pin also proves we are not racing eventpoll_release_file() on this epi, so drop the now-redundant re-check of epi->dying under f_lock. The cheap lockless READ_ONCE(epi->dying) fast-path bailout stays.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-46279	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: mm/alloc_tag: clear codetag for pages allocated before page_ext initialization Due to initialization ordering, page_ext is allocated and initialized relatively late during boot. Some pages have already been allocated and freed before page_ext becomes available, leaving their codetag uninitialized. A clear example is in init_section_page_ext(): alloc_page_ext() calls kmemleak_alloc(). If the slab cache has no free objects, it falls back to the buddy allocator to allocate memory. However, at this point page_ext is not yet fully initialized, so these newly allocated pages have no codetag set. These pages may later be reclaimed by KASAN, which causes the warning to trigger when they are freed because their codetag ref is still empty. Use a global array to track pages allocated before page_ext is fully initialized. The array size is fixed at 8192 entries, and will emit a warning if this limit is exceeded. When page_ext initialization completes, set their codetag to empty to avoid warnings when they are freed later. This warning is only observed with CONFIG_MEM_ALLOC_PROFILING_DEBUG=Y and mem_profiling_compressed disabled: [9.582133] ------------[cut here]------------ [9.582137] alloc_tag was not set [9.582139] WARNING: ./include/linux/alloc_tag.h:164 at __pgalloc_tag_sub+0x40f/0x550, CPU#5: systemd/1 [9.582190] CPU: 5 UID: 0 PID: 1 Comm: systemd Not tainted 7.0.0-rc4 #1 PREEMPT(lazy) [9.582192] Hardware name: Red Hat KVM, BIOS rel-1.16.3-0-ga6ed6b701f0a-prebuilt.qemu.org 04/01/2014 [9.582194] RIP: 0010:__pgalloc_tag_sub+0x40f/0x550 [9.582196] Code: 00 00 4c 29 e5 48 8b 05 1f 88 56 05 48 8d 4c ad 00 48 8d 2c c8 e9 87 fd ff ff 0f 0b 0f 0b e9 f3 fe ff ff 48 8d 3d 61 2f ed 03 <67> 48 0f b9 3a e9 b3 fd ff ff 0f 0b eb e4 e8 5e cd 14 02 4c 89 c7 [9.582197] RSP: 0018:ffffc9000001f940 EFLAGS: 00010246 [9.582200] RAX: dfffffc0000000000 RBX: 1ffff92000003f2b RCX: 1ffff110200d806c [9.582201] RDX: ffff8881006c0360 RSI: 0000000000000004 RDI: ffffffff9bc7b460 [9.582202] RBP: 0000000000000000 R08: 0000000000000000 R09: fffffbfff3a62324 [9.582203] R10: ffffffff9d311923 R11: 0000000000000000 R12: ffffea0004001b00 [9.582204] R13: 0000000000002000 R14: ffffea0000000000 R15: ffff8881006c0360 [9.582206] FS: 00007ffbbcf2d940(0000) GS:ffff888450479000(0000) knlGS:0000000000000000 [9.582208] CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033 [9.582210] CR2: 000055ee3aa260d0 CR3: 0000000148b67005 CR4: 0000000000770ef0 [9.582211] PKRU: 55555554 [9.582212] Call Trace: [9.582213] [9.582214] ? __pfx___pgalloc_tag_sub+0x10/0x10 [9.582216] ? check_bytes_and_report+0x68/0x140 [9.582219] __free_frozen_pages+0x2e4/0x1150 [9.582221] ? __free_slab+0xc2/0x2b0 [9.582224] qlist_free_all+0x4c/0xf0 [9.582227] kasan_quarantine_reduce+0x15d/0x180 [9.582229] __kasan_slab_alloc+0x69/0x90 [9.582232] kmem_cache_alloc_noprof+0x14a/0x500 [9.582234] do_getname+0x96/0x310 [9.582237] do_readlinkat+0x91/0x2f0 [9.582239] ? __pfx_do_readlinkat+0x10/0x10 [9.582240] ? get_random_bytes_user+0x1df/0x2c0 [9.582244] __x64_sys_readlinkat+0x96/0x100 [9.582246] do_syscall_64+0xce/0x650 [9.582250] ? __x64_sys_getrandom+0x13a/0x1e0 [9.582252] ? __pfx___x64_sys_getrandom+0x10/0x10 [9.582254] ? do_syscall_64+0x114/0x650 [9.582255] ? ksys_read+0xfc/0x1d0 [9.582258] ? __pfx_ksys_read+0x10/0x10 [9.582260] ? do_syscall_64+0x114/0x650 [9.582262] ? do_syscall_64+0x114/0x650 [9.582264] ? __pfx_fput_close_sync+0x10/0x10 [9.582266] ? file_close_fd_locked+0x178/0x2a0 [9.582268] ? __x64_sys_faccessat2+0x96/0x100 [9.582269] ? __x64_sys_close+0x7d/0xd0 [9.582271] ?

CVE	Severity	CVSS	CVSS Vector	Component	Description
					do_syscall_64+0x114/0x650 [9.582273] ? do_syscall_64+0x114/0x650 [9.582275] ? clear_bhb_loop+0x50/0xa0 [9.582277] ? clear_bhb_l ---truncated---
CVE-2026-46317	HIGH	8.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: KVM: arm64: Reassign nested_mmus array behind mmu_lock kvm->arch.nested_mmus[] is walked under kvm->mmu_lock, including from the MMU notifier path (kvm_unmap_gfn_range() -> kvm_nested_s2_unmap()), which can run at any time. kvm_vcpu_init_nested() reallocates the array and frees the old buffer while holding only kvm->arch.config_lock, so such a walker can reference the freed array. Allocate the new array outside of mmu_lock, as the allocation can sleep. Under the lock, copy the existing entries, fix up the back pointers and reassign the array. Free the old buffer after dropping the lock, as kvfree() can sleep as well.
CVE-2026-46324	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: nf_tables: use list_del_rcu for netlink hooks nft_netdev_unregister_hooks and __nft_unregister_flowtable_net_hooks need to use list_del_rcu(), this list can be walked by concurrent dumpers. Add a new helper and use it consistently.
CVE-2026-11824	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H	sqlite3 3.45.3	SQLite before 3.53.2 contains a heap-based buffer overflow vulnerability in the FTS5 full-text search extension that allows attackers to cause a crash or execute arbitrary code by supplying a crafted database with malicious continuation page metadata specifying a szLeaf value smaller than 4. Attackers can trigger an integer underflow in fts5ChunkIterate() causing an inflated remaining byte count during FTS5 MATCH query processing, leading to a heap buffer overflow of attacker-controlled data in applications compiled with SQLITE_ENABLE_FTS5.
CVE-2026-52953	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: iommu/vt-d: Fix oops due to out of scope access Below oops triggers when kill QEMU process: Oops: general protection fault, probably for non-canonical address 0x7fffffff844eaaa7: 0000 [#1] SMP NOPTI Call Trace: do_raw_spin_lock+0xaa/0xc0 _raw_spin_lock_irqsave+0x21/0x40 domain_remove_dev_pasid+0x52/0x160 intel_nested_set_dev_pasid+0x1b9/0x1e0 __iommu_set_group_pasid+0x56/0x120 pci_dev_reset_iommu_done+0xe3/0x180 pcie_flr+0x65/0x160 __pci_reset_function_locked+0x5b/0x120 vfio_pci_core_close_device+0x63/0xe0 [vfio_pci_core] vfio_df_close+0x4f/0xa0 vfio_df_unbind_iommufd+0x2d/0x60 vfio_device_fops_release+0x3e/0x40 __fput+0xe5/0x2c0 task_work_run+0x58/0xa0 do_exit+0x2c8/0x600 do_group_exit+0x2f/0xa0 get_signal+0x863/0x8c0 arch_do_signal_or_restart+0x24/0x100 exit_to_user_mode_loop+0x87/0x380 do_syscall_64+0x2ff/0x11e0 entry_SYSCALL_64_after_hwframe+0x76/0x7e The global static blocked domain is a dummy domain without corresponding dmar_domain structure, accessing beyond iommu_domain structure triggers oops easily. Fix it by return early in domain_remove_dev_pasid() like identity domain.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-52983	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: airoha: fix BQL imbalance in TX path Fix a possible BQL imbalance in airoha_dev_xmit(), where inflight packets are accounted only for the AIROHA_NUM_TX_RING netdev TX queues. The queue index is computed as: qid = skb_get_queue_mapping(skb) % ARRAY_SIZE(qdma->q_tx) txq = netdev_get_tx_queue(dev, qid); However, airoha_qdma_tx_napi_poll() accounts completions across all netdev TX queues (num_tx_queues), leading to inconsistent BQL accounting. Also reset all netdev TX queues in the ndo_stop callback.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-52991	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: sched/psi: fix race between file release and pressure write A potential race condition exists between pressure write and cgroup file release regarding the priv member of struct kernfs_open_file, which triggers the uaf reported in [1]. Consider the following scenario involving execution on two separate CPUs: CPU0 CPU1 ==== ==== vfs_rmdir() kernfs_iop_rmdir() cgroup_rmdir() cgroup_kn_lock_live() cgroup_destroy_locked() cgroup_addrm_files() cgroup_rm_file() kernfs_remove_by_name() kernfs_remove_by_name_ns() vfs_write() __kernfs_remove() new_sync_write() kernfs_drain() kernfs_fop_write_iter() kernfs_drain_open_files() cgroup_file_write() kernfs_release_file() pressure_write() cgroup_file_release() ctx = of->priv; kfree(ctx); of->priv = NULL; cgroup_kn_unlock() cgroup_kn_lock_live() cgroup_get(cgrp) cgroup_kn_unlock() if (ctx->psi.trigger) // here, trigger uaf for ctx, that is of->priv The cgroup_rmdir() is protected by the cgroup_mutex, it also safeguards the memory deallocation of of->priv performed within cgroup_file_release(). However, the operations involving of->priv executed within pressure_write() are not entirely covered by the protection of cgroup_mutex. Consequently, if the code in pressure_write(), specifically the section handling the ctx variable executes after cgroup_file_release() has completed, a uaf vulnerability involving of->priv is triggered. Therefore, the issue can be resolved by extending the scope of the cgroup_mutex lock within pressure_write() to encompass all code paths involving of->priv, thereby properly synchronizing the race condition occurring between cgroup_file_release() and pressure_write(). And, if an live kn lock can be successfully acquired while executing the pressure write operation, it indicates that the cgroup deletion process has not yet reached its final stage; consequently, the priv pointer within open_file cannot be NULL. Therefore, the operation to retrieve the ctx value must be moved to a point *after* the live kn lock has been successfully acquired. In another situation, specifically after entering cgroup_kn_lock_live() but before acquiring cgroup_mutex, there exists a different class of race condition: CPU0: write memory.pressure CPU1: write cgroup.pressure=0 ========== ========== kernfs_fop_write_iter() kernfs_get_active_of(of) pressure_write() cgroup_kn_lock_live(memory.pressure) cgroup_tryget(cgrp) kernfs_break_active_protection(kn) ... blocks on cgroup_mutex cgroup_pressure_write() cgroup_kn_lock_live(cgroup.pressure) cgroup_file_show(memory.pressure, false) kernfs_show(false) kernfs_drain_open_files() cgroup_file_release(of) kfree(ctx) of->priv = NULL cgroup_kn_unlock() ... acquires cgroup_mutex ctx = of->priv; // may now be NULL if (ctx->psi.trigger) // NULL dereference Consequently, there is a possibility that of->priv is NULL, the pressure write needs to check for this. Now that the scope of the cgroup_mutex has been expanded, the original explicit cgroup_get/put operations are no longer necessary, this is because acquiring/releasing the live kn lock inherently executes a cgroup get/put operation. [1] BUG: KASAN: slab-use-after-free in pressure_write+0xa4/0x210 kernel/cgroup/cgroup.c:4011 Call Trace: pressure_write+0xa4/0x210 kernel/cgroup/cgroup.c:4011 cgroup_file_write+0x36f/0x790 kernel/cgroup/cgroup.c:43 ---truncated---

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-53000	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: nat: use kfree_rcu to release ops Florian Westphal says: "Historically this is not an issue, even for normal base hooks: the data path doesn't use the original nf_hook_ops that are used to register the callbacks. However, in v5.14 I added the ability to dump the active netfilter hooks from userspace. This code will peek back into the nf_hook_ops that are available at the tail of the pointer-array blob used by the datapath. The nat hooks are special, because they are called indirectly from the central nat dispatcher hook. They are currently invisible to the nfnl hook dump subsystem though. But once that changes the nat ops structures have to be deferred too." Update nf_nat_register_fn() to deal with partial exposition of the hooks from error path which can be also an issue for nfnetlink_hook.
CVE-2026-53092	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: Fix linked reg delta tracking when src_reg == dst_reg Consider the case of rX += rX where src_reg and dst_reg are pointers to the same bpf_reg_state in adjust_reg_min_max_vals(). The latter first modifies the dst_reg in-place, and later in the delta tracking, the subsequent is_reg_const(src_reg)/reg_const_value(src_reg) reads the post-{add,sub} value instead of the original source. This is problematic since it sets an incorrect delta, which sync_linked_regs() then propagates to linked registers, thus creating a verifier-vs-runtime mismatch. Fix it by just skipping this corner case.
CVE-2026-53109	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: powerpc/pgtable-frag: Fix bad page state in pte_frag_destroy powerpc uses pt_frag_refcount as a reference counter for tracking it's pte and pmd page table fragments. For PTE table, in case of Hash with 64K pagesize, we have 16 fragments of 4K size in one 64K page. Patch series [1] "mm: free retracted page table by RCU" added pte_free_defer() to defer the freeing of PTE tables when retract_page_tables() is called for madvise MADV_COLLAPSE on shmem range. [1]: https://lore.kernel.org/all/7cd843a9-aa80-14f-5eb2-33427363c20@google.com/ pte_free_defer() sets the active flag on the corresponding fragment's folio & calls pte_fragment_free(), which reduces the pt_frag_refcount. When pt_frag_refcount reaches 0 (no active fragment using the folio), it checks if the folio active flag is set, if set, it calls call_rcu to free the folio, it the active flag is unset then it calls pte_free_now(). Now, this can lead to following problem in a corner case... [265.351553][T183] BUG: Bad page state in process a.out pfn:20d62 [265.353555][T183] page: refcount:0 mapcount:0 mapping:0000000000000000 index:0x0 pfn:0x20d62 [265.355457][T183] flags: 0x3ffff800000100(active|node=0|zone=0|lastcpupid=0x7ffff) [265.358719][T183] raw: 003ffff800000100 0000000000000000 5deadbeef0000122 0000000000000000 [265.360177][T183] raw: 0000000000000000 c0000000119caf58 00000000ffffffff 0000000000000000 [265.361438][T183] page dumped because: PAGE_FLAGS_CHECK_AT_FREE flag(s) set [265.362572][T183] Modules linked in: [265.364622][T183] CPU: 0 UID: 0 PID: 183 Comm: a.out Not tainted 6.18.0-rc3-00141-g1ddeaaace7ff-dirty #53 VOLUNTARY [265.364785][T183] Hardware name: IBM pSeries (emulated by qemu) POWER10 (architected) 0x801200 0xf000006 of:SLOF,git-ee03ae pSeries [265.364908][T183] Call Trace: [265.364955][T183] [c000000011e6f7c0] [c000000001cfaa18] dump_stack_lvl+0x130/0x148 (unreliable) [265.365202][T183] [c000000011e6f7f0] [c000000000794758] bad_page+0xb4/0x1c8 [

CVE	Severity	CVSS	CVSS Vector	Component	Description
					265.365384][T183] [c000000011e6f890] [c00000000079c020] __free_frozen_pages+0x838/0xd08 [265.365554][T183] [c000000011e6f980] [c0000000000a70ac] pte_frag_destroy+0x298/0x310 [265.365729][T183] [c000000011e6fa30] [c0000000000aa764] arch_exit_mmap+0x34/0x218 [265.365912][T183] [c000000011e6fa80] [c000000000751698] exit_mmap+0xb8/0x820 [265.366080][T183] [c000000011e6fc30] [c0000000001b1258] __mmput+0x98/0x300 [265.366244][T183] [c000000011e6fc80] [c0000000001c81f8] do_exit+0x470/0x1508 [265.366421][T183] [c000000011e6fd70] [c0000000001c95e4] do_group_exit+0x88/0x148 [265.366602][T183] [c000000011e6fdc0] [c0000000001c96ec] pid_child_should_wake+0x0/0x178 [265.366780][T183] [c000000011e6fdf0] [c00000000003a270] system_call_exception+0x1b0/0x4e0 [265.366958][T183] [c000000011e6fe50] [c00000000000d05c] system_call_vectored_common+0x15c/0x2ec The bad page state error occurs when such a folio gets freed (with active flag set), from do_exit() path in parallel. ... this can happen when the pte fragment was allocated from this folio, but when all the fragments get freed, the pte_frag_refcount still had some unused fragments. Now, if this process exits, with such folio as it's cached pte_frag in mm->context, then during pte_frag_destroy(), we simply call pagetable_dtor() and pagetable_free(), meaning it doesn't clear the active flag. This, can lead to the above bug. Since we are anyway in do_exit() path, then if the refcount is 0, then I guess it should be ok to simply clear the folio active flag before calling pagetable_dtor() & pagetable_free().
CVE-2026-53118	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: vdpa: use generic driver_override infrastructure When a driver is probed through __driver_attach(), the bus' match() callback is called without the device lock held, thus accessing the driver_override field without a lock, which can cause a UAF. Fix this by using the driver-core driver_override infrastructure taking care of proper locking internally. Note that calling match() from __driver_attach() without the device lock held is intentional. [1]
CVE-2026-53179	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: staging: rtl8723bs: fix buffer over-read in rtw_update_protection rtw_update_protection() is called with a pointer offset into the ies buffer but the full ie_length is passed, causing a potential buffer over-read.
CVE-2026-53250	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: xsk: cache csum_start/csum_offset to fix TOCTOU in xsk_skb_metadata() The TX metadata area resides in the UMEM buffer which is memory-mapped and concurrently writable by userspace. In xsk_skb_metadata(), csum_start and csum_offset are read from shared memory for bounds validation, then read again for skb assignment. A malicious userspace application can race to overwrite these values between the two reads, bypassing the bounds check and causing out-of-bounds memory access during checksum computation in the transmit path. Fix this by reading csum_start and csum_offset into local variables once, then using the local copies for both validation and assignment. Note that other metadata fields (flags, launch_time) and the cached csum fields may be mutually inconsistent due to concurrent userspace writes, but this is benign: the only security-critical invariant is that each field's validated value is the same one used, which local caching guarantees.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-53277	HIGH	8.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: KVM: arm64: Take the SRCU lock for page table walks in fault injection and AT emulation walk_s1() and kvm_walk_nested_s2() expect to be called while holding kvm->srcu to guard against memslot changes. While this is generally the case, __kvm_at_s12() and __kvm_find_s1_desc_level() call into the respective walkers without taking kvm->srcu. Fix by acquiring kvm->srcu prior to the table walk in both instances.
CVE-2026-53281	HIGH	8.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: iommu/vt-d: Avoid NULL pointer dereference or refcount corruption Commit 60f030f7418d ("iommu/vt-d: Avoid use of NULL after WARN_ON_ONCE") fixed a NULL pointer dereference in an unlikely situation partly. If dev_pasid is not found in the dev_pasids list, it remains NULL. However, the teardown operations are executed unconditionally, this lead to a NULL pointer dereference or refcount corruption. If the domain was never attached to this IOMMU, info will be NULL, which would cause an immediate dereference when checking --info->refcnt. Even if info is not NULL, decrementing the refcount without having removed a valid PASID might unbalance the count. This could lead to premature dropping of the refcount to 0, potentially causing a use-after-free for the remaining active devices sharing the domain. Fix it by returning early if dev_pasid is NULL, before executing the teardown operations. Issue found by AI review and suggested by Kevin Tian. https://sashiko.dev/#/patchset/20260421031347.1408890-1-zhenzhong.duan%40intel.com
CVE-2026-53284	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: btrfs: only release the dirty pages io tree after successful writes [WARNING] With extra warning on dirty extent buffers at umount (aka, the next patch in the series), test case generic/388 can trigger the following warning about dirty extent buffers at unmount time: BTRFS critical (device dm-2 state E): emergency shutdown BTRFS error (device dm-2 state E): error while writing out transaction: -30 BTRFS warning (device dm-2 state E): Skipping commit of aborted transaction. BTRFS error (device dm-2 state EA): Transaction 9 aborted (error -30) BTRFS: error (device dm-2 state EA) in cleanup_transaction:2068: errno=-30 Readonly filesystem BTRFS info (device dm-2 state EA): forced readonly BTRFS info (device dm-2 state EA): last unmount of filesystem 4fbf2e15-f941-49a0-bc7c-716315d2777c ------------[cut here]------------ WARNING: disk-io.c:3311 at invalidate_and_check_btree_folios+0xfd/0x1ca [btrfs], CPU#8: umount/914368 CPU: 8 UID: 0 PID: 914368 Comm: umount Tainted: G OE 7.1.0-rc1-custom+ #372 PREEMPT(full) 2de38db8d1deae71fde295430a0ff3ab98ccf596 Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS unknown 02/02/2022 RIP: 0010:invalidate_and_check_btree_folios+0xfd/0x1ca [btrfs] Call Trace: close_ctree+0x52e/0x574 [btrfs d2f0b1cd330d1287e7a9919d112eadfc0e914efd] generic_shutdown_super+0x89/0x1a0 kill_anon_super+0x16/0x40 btrfs_kill_super+0x16/0x20 [btrfs d2f0b1cd330d1287e7a9919d112eadfc0e914efd] deactivate_locked_super+0x2d/0xb0 cleanup_mnt+0xdc/0x140 task_work_run+0x5a/0xa0 exit_to_user_mode_loop+0x123/0x4b0 do_syscall_64+0x243/0x7c0 entry_SYSCALL_64_after_hwframe+0x4b/0x53 ---[end trace 0000000000000000]--- BTRFS warning (device dm-2 state EA): unable to release extent buffer 30539776 owner 9 gen 9 refs 2 flags 0x7 BTRFS warning (device dm-2 state EA): unable to release extent buffer

CVE	Severity	CVSS	CVSS Vector	Component	Description
					30621696 owner 257 gen 9 refs 2 flags 0x7 BTRFS warning (device dm-2 state EA): unable to release extent buffer 30638080 owner 258 gen 9 refs 2 flags 0x7 BTRFS warning (device dm-2 state EA): unable to release extent buffer 30654464 owner 7 gen 9 refs 2 flags 0x7 BTRFS warning (device dm-2 state EA): unable to release extent buffer 30703616 owner 2 gen 9 refs 2 flags 0x7 BTRFS warning (device dm-2 state EA): unable to release extent buffer 30720000 owner 10 gen 9 refs 2 flags 0x7 BTRFS warning (device dm-2 state EA): unable to release extent buffer 30736384 owner 4 gen 9 refs 2 flags 0x7 BTRFS warning (device dm-2 state EA): unable to release extent buffer 30752768 owner 11 gen 9 refs 2 flags 0x7 I'm using a stripped down version, which seems to trigger the warning more reliably: _fsstress_pid="" workload() { dmesg -C mkfs.btrfs -f -K $dev > /dev/null echo 1 > /sys/kernel/debug/clear_warn_once mount $dev $mnt $fsstress -w -n 1024 -p 4 -d $mnt & _fsstress_pid=$! sleep 0 $godown $mnt pkill -- echo -PIPE fsstress > /dev/null wait $_fsstress_pid unset _fsstress_pid umount $mnt if dmesg | grep -q "WARNING"; then fail fi } for ((i = 0; i < $runtime; i++)); do echo "=== $i/$runtime ===" workload done [CAUSE] Inside btrfs_write_and_wait_transaction(), we first try to write all dirty ebs, then wait for them to finish. After that we call btrfs_extent_io_tree_release() to free all extent states from dirty_pages io tree. However if we hit an error from btrfs_write_marked_extent(), then we still call btrfs_extent_io_tree_release() to clear that dirty_pages io tree, which may contain dirty records that we haven't yet submitted. Furthermore, the later transaction cleanup path will utilize that dirty_pages io tree to properly cleanup those dirty ebs, but since it's already empty, no dirty ebs are properly cleaned up, thus will later trigger the warnings inside invalidate_btree_folios(). ---truncated---
CVE-2026-58014	HIGH	7.3	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L	glib-2.0 2.78.6	A flaw was found in GLib. An off-by-one error can occur in the g_key_file_get_locale_string_list function in the gkeyfile.c file when loading a key file with an empty value. This flaw can cause an out-of-bounds access of 1 byte or a denial of service when the out-of-bounds access crosses a page boundary.
CVE-2026-58016	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	glib-2.0 2.78.6	A flaw was found in GLib. A state confusion issue exists in g_dbus_node_info_new_for_xml() in the gio/gdbusintrospection.c file when processing malformed D-Bus introspection XML, specifically with a `node` element nested within other elements like `method`, `signal`, `property` or `arg`. This issue can cause an unsigned integer overflow and lead to an out-of-bounds read, resulting in a denial of service.
CVE-2026-53330	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/amd/display: Fix out-of-bounds read in dp_get_eq_aux_rd_interval() [Why & How] The aux_rd_interval array in struct dc_lttpr_caps is declared with MAX_REPEATER_CNT - 1 (7) elements, indexed 0..6. However, the offset parameter passed to dp_get_eq_aux_rd_interval() can be as large as MAX_REPEATER_CNT (8) when a sink reports 8 LTTPR repeaters via DPCD. This leads to an out-of-bounds read of aux_rd_interval[7] when offset is 8. Fix this by growing aux_rd_interval to MAX_REPEATER_CNT elements to accommodate the full range of valid repeater counts defined by the DP spec. (cherry picked from commit a55a458a8df37a65ffda5cf721d554a8f74f6b04)
CVE-2026-60002	HIGH	7.7	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L	openssh 9.6p1	ssh in OpenSSH before 10.4 can have a use-after-free when a server changes its host key during a key re-exchange. (This outcome occurs only on the client side.)

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-53368	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: f2fs: fix fsck inconsistency caused by incorrect nat_entry flag usage f2fs_need_dentry_mark() reads nat_entry flags without mutual exclusion with the checkpoint path, which can result in an incorrect inode block marking state. The scenario is as follows: create & write & fsync 'file A' write checkpoint - f2fs_do_sync_file // inline inode - f2fs_write_inode // inode folio is dirty - f2fs_write_checkpoint - f2fs_flush_merged_writes - f2fs_sync_node_pages - f2fs_fsync_node_pages // no dirty node - f2fs_need_inode_block_update // return true - f2fs_fsync_node_pages // inode dirtied - f2fs_need_dentry_mark //return true - f2fs_flush_nat_entries - f2fs_write_checkpoint end - __write_node_folio // inode with DENT_BIT_SHIFT set SPO, "fsck --dry-run" find inode has already checkpointed but still with DENT_BIT_SHIFT set The state observed by f2fs_need_dentry_mark() can differ from the state observed in __write_node_folio() after acquiring sbi->node_write. The root cause is that the semantics of IS_CHECKPOINTED and HAS_FSYNCED_INODE are only guaranteed after the checkpoint write has fully completed. This patch moves set_dentry_mark() into __write_node_folio() and protects it with the sbi->node_write lock.
CVE-2026-63805	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: crypto: nx - fix nx_crypto_ctx_exit argument nx_crypto_ctx_shash_exit calls nx_crypto_ctx_exit with crypto_shash_ctx(...) but crypto_shash_ctx gives a nx_crypto_ctx *, not a crypto_tfm *. Fix the type in nx_crypto_ctx_exit and drop the bogus crypto_tfm_ctx call. This fixes the following oops: BUG: Unable to handle kernel data access at 0xc0403effffffffc8 Faulting instruction address: 0xc000000000396cb4 Oops: Kernel access of bad area, sig: 11 [#15] Call Trace: nx_crypto_ctx_shash_exit+0x24/0x60 crypto_shash_exit_tfm+0x28/0x40 crypto_destroy_tfm+0x98/0x140 crypto_exit_ahash_using_shash+0x20/0x40 crypto_destroy_tfm+0x98/0x140 hash_release+0x1c/0x30 alg_sock_destruct+0x38/0x60 __sk_destruct+0x48/0x2b0 af_alg_release+0x58/0xb0 __sock_release+0x68/0x150 sock_close+0x20/0x40 __fput+0x110/0x3a0 sys_close+0x48/0xa0 system_call_exception+0x140/0x2d0 system_call_common+0xf4/0x258 .. which came from hardlink(1) opportunistically using AF_ALG. The same problem exists with nx_crypto_ctx_skcipher_exit getting a context it wasn't expecting, but apparently nobody hit that for years.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-63819	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: f2fs: fix to do sanity check on f2fs_get_node_folio_ra() kernel BUG at fs/f2fs/file.c:845! Oops: invalid opcode: 0000 [#1] SMP KASAN NOPTI CPU: 0 UID: 0 PID: 5336 Comm: syz.0.0 Not tainted syzkaller #0 PREEMPT(full) Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.16.3-debian-1.16.3-2 04/01/2014 RIP: 0010:f2fs_do_truncate_blocks+0x1115/0x1140 fs/f2fs/file.c:845 Code: fc fc 90 0f 0b e8 8b 9d 9a fd 90 0f 0b e8 83 9d 9a fd 48 89 df 48 c7 c6 60 d1 1a 8c e8 54 f1 fc fc 90 0f 0b e8 6c 9d 9a fd 90 <0f> 0b e8 64 9d 9a fd 90 0f 0b 90 e9 93 fd ff ff e8 56 9d 9a fd 90 RSP: 0018:ffffc9000e4474c0 EFLAGS: 00010283 RAX: ffffffff842b1d34 RBX: 0000000000000003 RCX: 0000000000100000 RDX: ffffc9000f03a000 RSI: 0000000000035503 RDI: 0000000000035504 RBP: ffffc9000e447608 R08: ffff8880123b0000 R09: 0000000000000002 R10: 00000000ffffffe R11: 0000000000000002 R12: 0000000000000001 R13: 0000000000000000 R14: 1ffff92001c88ea0 R15: 00000000ffff039c FS: 00007f7e02ee36c0(0000) GS:ffff88808c887000(0000) knlGS:0000000000000000 CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033 CR2: 00007ff0305c4000 CR3: 0000000012d4c000 CR4: 0000000000352ef0 Call Trace: f2fs_truncate_blocks+0x10a/0x300 fs/f2fs/file.c:882 f2fs_truncate+0x471/0x7c0 fs/f2fs/file.c:940 f2fs_evict_inode+0xa3f/0x1ac0 fs/f2fs/inode.c:907 evict+0x61e/0xb10 fs/inode.c:841 f2fs_fill_super+0x5f43/0x78f0 fs/f2fs/super.c:5224 get_tree_bdev_flags+0x431/0x4f0 fs/super.c:1694 vfs_get_tree+0x92/0x2a0 fs/super.c:1754 fc_mount fs/namespace.c:1193 [inline] do_new_mount_fc fs/namespace.c:3758 [inline] do_new_mount+0x341/0xd30 fs/namespace.c:3834 do_mount fs/namespace.c:4167 [inline] __do_sys_mount fs/namespace.c:4383 [inline] __se_sys_mount+0x31d/0x420 fs/namespace.c:4360 do_syscall_x64 arch/x86/entry/syscall_64.c:63 [inline] do_syscall_64+0x15f/0xf80 arch/x86/entry/syscall_64.c:94 entry_SYSCALL_64_after_hwframe+0x77/0x7f count = ADDRS_PER_PAGE(dn.node_folio, inode); count -= dn.ofs_in_node; f2fs_bug_on(sbi, count < 0); The fuzz test will trigger above bug_on in f2fs. The root cause should be: in the corrupted inode, there is a direct node which has the same ino and nid in its footer, so in f2fs_do_truncate_blocks(), after f2fs_get_dnode_of_data() finds such dnode: 1) ADDRS_PER_PAGE(dn.node_folio, inode) will return 923 2) once dn.ofs_in_node points to addr[923, 1017] Then it will trigger the system panic. Let's introduce NODE_TYPE_NON_IXNODE to indicate current node should not be an inode or xattr node, and then use it in below path to detect inconsistent node chain in inode mapping table: - f2fs_do_truncate_blocks - f2fs_get_dnode_of_data - f2fs_get_node_folio_ra - __get_node_folio - f2fs_sanity_check_node_footer - case NODE_TYPE_NON_IXNODE -> check whether it is inode|xnode

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-63832	HIGH	8.8	CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: mt76: add wcid publish check in mt76_sta_add Since mt7925_mac_sta_add publishes wcid, add publish check in mt76_sta_add to avoid reinitializing the wcid->poll_list. Found dev->sta_poll_list corruption when using mt7925 and 7.1-rc4. According to the corruption information, prev->next was changed to itself. wlan0: disconnect from AP 90:fb:5d:94:8b:e3 for new auth to 90:fb:5d:94:8b:e2 wlan0: authenticate with 90:fb:5d:94:8b:e2 (local address=84:9e:56:9c:7e:6b) wlan0: send auth to 90:fb:5d:94:8b:e2 (try 1/3) slab kmalloc-8k start ffff8c80958a6000 pointer offset 4160 size 8192 list_add corruption. prev->next should be next (ffff8c808a7488f8), but was ffff8c80958a7040. (prev=ffff8c80958a7040). mt76_wcid_add_poll+0x95/0xd0 [mt76] mt7925_mac_add_txs.part.0+0xa5/0xe0 [mt7925_common] mt7925_rx_check+0xa7/0xc0 [mt7925_common] mt76_dma_rx_poll+0x50d/0x790 [mt76] mt792x_poll_rx+0x52/0xe0 [mt792x_lib]
CVE-2026-63853	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/amdgpu/vcn: set no_user_fence for VCN v4.0 enc ring VCN encoder and decoder rings do not support 64-bit user fence writes, reject CS submissions with user fences. (cherry picked from commit fd852c048b46f9825e904a4f3f4538fe9d8827d9)
CVE-2026-63858	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: nf_tables: add hook transactions for device deletions Restore the flag that indicates that the hook is going away, ie. NFT_HOOK_REMOVE, but add a new transaction object to track deletion of hooks without altering the basechain/flowtable hook_list during the preparation phase. The existing approach that moves the hook from the basechain/flowtable hook_list to transaction hook_list breaks netlink dump path readers of this RCU-protected list. It should be possible use an array for nft_trans_hook to store the deleted hooks to compact the representation but I am not expecting many hook object, specially now that wildcard support for devices is in place. Note that the nft_trans_chain_hooks() list contains a list of struct nft_trans_hook objects for DELCHAIN and DELFLOWTABLE commands, while this list stores struct nft_hook objects for NEWCHAIN and NEWFLOWTABLE. Note that new commands can be updated to use nft_trans_hook for consistency. This patch also adapts the event notification path to deal with the list of hook transactions.
CVE-2026-63879	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/amdgpu: fix amdgpu_hmm_range_get_pages The notifier sequence must only be read once or otherwise we could work with invalid pages. While at it also fix the coding style, e.g. drop the pre-initialized return value and use the common define for 2G range. (cherry picked from commit c08972f555945cda57b0adb72272a37910153390)

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-63941	HIGH	8.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: KVM: arm64: Correctly cap ZCR_EL2 provided by a guest hypervisor ZCR_EL2 can be updated by a VHE guest hypervisor either using ZCR_EL2 (which traps) or ZCR_EL1 (which does not trap). KVM handles both in different way: - on ZCR_EL2 trap, ZCR_EL2.LEN is immediately capped at the VM's own VL limit. This has the potential to break existing SW that relies on the full LEN field to be stateful. - on ZCR_EL1 access, we do absolutely nothing. On restoring the SVE context for an L2 guest, we directly restore the guest hypervisor's view of ZCR_EL2 into the physical ZCR_EL2. If the guest's view of the register was updated using the ZCR_EL2 accessor, the value has already been sanitised (with the caveat mentioned above). But if the guest used ZCR_EL1, the raw value is written into the HW, and the L2 guest can now access VLs that it shouldn't. Fix all the above by moving the VL capping to the restore points, ensuring that: - the HW is always programmed with a capped value, irrespective of the accessor being used, - the ZCR_EL2.LEN field is always completely stateful, irrespective of the accessor being used. Additionally, move ZCR_EL2 to be a sanitised register, ensuring that only the LEN field is actually stateful. This requires some creative construction of the RES0 mask, as the sysreg generation script does not yet generate RAZ/WI fields. [maz: rewrote commit message, tidy up access_zcr_el2()]
CVE-2026-64017	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: blk-mq: pop cached request if it is usable When submitting a bio to blk-mq, if the task should sleep after peeking a cached request, but before it pops it, the plug flushes and calls blk_mq_free_plug_rqs, freeing the cached_rqs. This creates a use-after-free bug. Fix this by popping the cached request before any possible blocking calls if it is suitable for use. Popping this request first holds a queue reference, so avoid any serialization races with queue freezes and can safely proceed with dispatching that request to the driver. This potentially increases a timing window from when a driver wants to freeze its queue to when requests stop being dispatched. That scenario is off the fast path though, and drivers need to appropriately handle requests during a freeze request anyway. The downside is the popped element needs to be individually freed when we performed a bio plug merge. The cached request would have had to be freed later anyway, but this patch does it inline with building the plug list instead of after flushing it.
CVE-2026-64036	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: cgroup/rstat: validate cpu before css_rstat_cpu() access css_rstat_updated() is exposed as a BPF kfunc and accepts a caller-provided cpu argument. The function uses cpu for per-cpu rstat lookups without checking whether it refers to a valid possible CPU. A BPF iter/cgroup program with CAP_BPF and CAP_PERFMON can pass an invalid cpu value. On an unfixed UBSCAN_BOUNDS test kernel, cpu == 0x7fffffff triggers: UBSAN: array-index-out-of-bounds in kernel/cgroup/rstat.c:31:9 index 2147483647 is out of range for type 'long unsigned int [64]' Call Trace: css_rstat_updated bpf_iter_run_prog cgroup_iter_seq_show bpf_seq_read Add cpu validation to the BPF-facing css_rstat_updated() kfunc and move the common implementation to __css_rstat_updated() for in-kernel callers.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-64058	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfs: Fix netfs_read_folio() to wait on writeback Fix netfs_read_folio() to wait for an ongoing writeback to complete so that it can trust the dirty flag and whatever is attached to folio->private (folio->private may get cleaned up by the collector before it clears the writeback flag).
CVE-2026-64076	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: bridge: eb_tables: close module init race sashiko reports for unrelated patch: Does the core ebtables initialization in ebtables.c suffer from a similar race? Once nf_register_sockopt() completes, the sockopts are exposed globally. sockopt has to be registered last, just like in ip/ip6/arptables.
CVE-2026-64077	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: ebtables: move to two-stage removal scheme Like previous patches for x_tables, follow same pattern in ebtables. We can't reuse xt helpers: ebt_table struct layout is incompatible. table->ops assignment is now done while still holding the ebt mutex to make sure we never expose partially-filled table struct.
CVE-2026-64078	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: x_tables: add and use xtables_unregister_table_exit Previous change added xtables_unregister_table_pre_exit to detach the table from the packetpath and to unlink it from the active table list. In case of rmmod, userspace that is doing set/getsockopt for this table will not be able to re-instantiate the table: 1. The larval table has been removed already 2. existing instantiated table is no longer on the xt pernet table list. This adds the second stage helper: unlink the table from the dying list, free the hook ops (if any) and do the audit notification. It replaces xt_unregister_table().
CVE-2026-64255	HIGH	8.8	CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: iwlwifi: mld: validate sta_mask before ffs() in BA session handlers Three BA session handlers use ffs(ba_data->sta_mask) - 1 to derive a station ID without checking that sta_mask is non-zero. When sta_mask is zero, ffs() returns 0 and the subtraction wraps to 0xFFFFFFFF, causing an out-of-bounds access on fw_id_to_link_sta[]. Add WARN_ON_ONCE(!ba_data->sta_mask) guards before each ffs() call, consistent with the existing check in iwl_mld_ampdu_rx_start().
CVE-2026-64280	HIGH	8.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: fpga: dfl-afu: validate DMA mapping length in afu_dma_map_region() afu_ioctl_dma_map() accepts a 64-bit length from userspace via DFL_FPGA_PORT_DMA_MAP ioctl without an upper bound check. The value is passed to afu_dma_pin_pages() where npages is derived as length >> PAGE_SHIFT and passed to pin_user_pages_fast() which takes int nr_pages, causing implicit truncation if length is very large. Validate map.length at the ioctl entry point before calling afu_dma_map_region(), rejecting values whose page count exceeds INT_MAX.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-64389	HIGH	8.2	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ksmbd: validate NTLMv2 response before updating session key ksmbd_auth_ntlmv2() derives the NTLMv2 session key into sess->sess_key before it verifies the NTLMv2 response. ksmbd_decode_ntlmssp_auth_blob() then continues into KEY_XCH even when ksmbd_auth_ntlmv2() failed. With SMB3 multichannel binding, the failed authentication operates on an existing session and the session setup error path does not expire binding sessions. A client can send a binding session setup with a bad NT proof and KEY_XCH and still modify sess->sess_key before STATUS_LOGON_FAILURE is returned. Relevant path: smb2_sess_setup() -> conn->binding = true -> ntlm_authenticate() -> session_user() -> ksmbd_decode_ntlmssp_auth_blob() -> ksmbd_auth_ntlmv2() -> calc_ntlmv2_hash() -> hmac_md5_usingrawkey(..., sess->sess_key) -> crypto_memneq() returns mismatch -> KEY_XCH arc4_crypt(..., sess->sess_key, ...) -> out_err without expiring the binding session Derive the base session key into a local buffer and copy it to sess->sess_key only after the proof matches. Return immediately on authentication failure so KEY_XCH is only processed after successful authentication.
CVE-2026-64520	HIGH	8.4	CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: firmware: arm_ffa: Bound PARTITION_INFO_GET_REGS copies The register-based PARTITION_INFO_GET path trusted the firmware-provided indices when copying partition descriptors into the caller buffer. Reject inconsistent counts or index progressions so the copy loop cannot write past the allocated array. (fixed cur_idx when exactly one descriptor in the first fragment)
CVE-2026-64561	HIGH	8.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: KVM: x86: Check for invalid/obsolete root *after* making MMU pages available Check for a "stale" page fault, i.e. for an invalid and/or obsolete root, after making MMU pages available for the shadow MMU. If reclaiming shadow pages zaps an in-use root, i.e. marks it invalid, then KVM will attempt to map memory into an invalid root. On its own, populating an invalid root is "fine", but because child shadow pages inherit their parent's role, any children created during the map/fetch will be created as invalid pages, thus violating KVM's invariant that invalid pages are never on the list of active MMU pages. Note, the underlying flaw has existed since KVM first started tracking invalid roots in 2008 (commit 2e53d63acba7, "KVM: MMU: ignore zapped root pagetables"), but the true badness only came along in 2020 (Linux 5.9) with the invariant that invalid shadow pages can't be on the list of active pages. Note #2, inheriting role.invalid when creating child shadow pages is also far from ideal; that flaw will be addressed separately.
CVE-2026-64562	HIGH	8.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: KVM: nVMX: Hide shadow VMCS right after VMCLEAR free_nested() frees the shadow VMCS while vmcs01 still points to it. But because it is asynchronous with respect to loaded_vmcs_clear(), the vCPU might migrate before the pointer is cleared and __loaded_vmcs_clear() may then execute VMCLEAR. The VMCS needs to stay attached until its explicit VMCLEAR completes, but then it can be hidden and the page safely freed.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-64563	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: rhashtable: clear stale iter->p on table restart rhashtable_walk_start_check() has two restart paths when resuming a walk. When iter->walker.tbl is valid, it re-validates iter->p against the table and sets iter->p = NULL if the object is gone. When iter->walker.tbl is NULL (table was freed during resize), it resets slot and skip but forgets to clear iter->p. rhashtable_walk_next() then dereferences the stale iter->p, reading freed memory. This is a use-after-free. Any caller that does multi-fragment rhashtable walks across walk_stop/walk_start boundaries is affected. Concrete cases include netlink_diag (__netlink_diag_dump in net/netlink/diag.c) and TIPC (tipc_nl_sk_walk in net/tipc/socket.c). Crash stack (netlink_diag): BUG: KASAN: slab-use-after-free in rhashtable_walk_next+0x365/0x3c0 Read of size 8 at addr ffff88801a9d2438 (freed kmalloc-2k, offset 1080) Call Trace: rhashtable_walk_next+0x365/0x3c0 (lib/rhashtable.c:1016) __netlink_diag_dump+0x160/0x760 (net/netlink/diag.c:122) netlink_diag_dump+0xc2/0x240 netlink_dump+0x5bc/0x1270 netlink_recvmsg+0x7a3/0x980 sock_recvmsg+0x1bc/0x200 __sys_recvfrom+0x1d4/0x2c0
CVE-2026-64565	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: Input: ims-pcu - fix heap-buffer-overflow in ims_pcu_process_data() The `ims_pcu_process_data()` processes incoming URB data byte by byte. However, it fails to check if the `read_pos` index exceeds IMS_PCU_BUF_SIZE. If a malicious USB device sends a packet larger than IMS_PCU_BUF_SIZE, `read_pos` will increment indefinitely. Moreover, since `read_pos` is located immediately after `read_buf`, the attacker can overwrite `read_pos` itself to arbitrarily control the index. This manipulated `read_pos` is subsequently used in `ims_pcu_handle_response()` to copy data into `cmd_buf`, leading to a heap buffer overflow. Specifically, an attacker can overwrite the `cmd_done.wait.head` located at offset 136 relative to `cmd_buf` in the `ims_pcu_handle_response()`. Consequently, when the driver calls `complete(&pcu->cmd_done)`, it triggers a control flow hijack by using the manipulated pointer. Fix this by adding a bounds check for `read_pos` before writing to `read_buf`. If the packet is too long, discard it, log a warning, and reset the parser state. [dtor: factor out resetting packet state, reset checksum as well]

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-64567	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: btrfs: reject free space cache with more entries than pages When loading a v1 free space cache, __load_free_space_cache() takes num_entries and num_bitmaps straight from the on-disk btrfs_free_space_header. That header is stored in the tree_root under a key with type 0, which the tree-checker has no case for, so neither count is validated before the load trusts it. The load loops num_entries times and maps the next page whenever the current one runs out, going through io_ctl_check_crc() -> io_ctl_map_page(), which does io_ctl->pages[io_ctl->index++]. But pages[] is allocated in io_ctl_init() from the cache inode's i_size, not from num_entries: num_pages = DIV_ROUND_UP(i_size_read(inode), PAGE_SIZE); io_ctl->pages = kcalloc(num_pages, sizeof(struct page *), GFP_NOFS); So if num_entries claims more records than the pages can hold, io_ctl->index runs off the end of pages[]. The write side never hits this because io_ctl_add_entry() and io_ctl_add_bitmap() both stop once io_ctl->index >= io_ctl->num_pages; the read side just never had the same check. To trigger it, take a clean cache (num_entries = here), set num_entries in the header to 0x10000, and fix up the leaf checksum so it still passes the tree-checker. The cache inode has i_size = 65536, so num_pages is 16 and pages[] is a 16-pointer (kmalloc-128) array. The load now tries to read 65536 entries, io_ctl->index walks up to 16, and pages[16] is read past the array. BUG: KASAN: slab-out-of-bounds in io_ctl_check_crc (fs/btrfs/free-space-cache.c:420 fs/btrfs/free-space-cache.c:565) Read of size 8 at addr ffff88800c833a80 by task kworker/u8:3/58 io_ctl_check_crc (fs/btrfs/free-space-cache.c:420 fs/btrfs/free-space-cache.c:565) __load_free_space_cache (fs/btrfs/free-space-cache.c:655 fs/btrfs/free-space-cache.c:820) load_free_space_cache (fs/btrfs/free-space-cache.c:1017) caching_thread (fs/btrfs/block-group.c:880) btrfs_work_helper (fs/btrfs/async-thread.c:312) process_one_work worker_thread kthread ret_from_fork free-space-cache.c:420 is io_ctl_map_page(), inlined into io_ctl_check_crc() at line 565, which is why that is the frame KASAN names. The out-of-bounds slot is then treated as a struct page and handed to crc32c(), so the bad read turns into a GP fault. Add the missing check to io_ctl_check_crc(), which is where both the entry loop and the bitmap loop end up. When num_entries is too large the load now fails like any corrupt cache: __load_free_space_cache() drops it and rebuilds the free space from the extent tree, so a valid cache is never rejected.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-64568	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: mac80211: fix unsol_bcast_probe_resp double free on alloc failure ieee80211_set_unsol_bcast_probe_resp() calls kfree_rcu() on the old template before allocating the replacement. If the kzalloc() then fails, it returns -ENOMEM while link->u.ap.unsol_bcast_probe_resp still points at the object already queued for freeing. A later update or AP teardown re-queues that same rcu_head; the second free is caught by KASAN when the RCU sheaf is processed in softirq: BUG: KASAN: double-free in rcu_free_sheaf (mm/slub.c:5850) Free of addr ffff88800d06f300 by task exploit/145 ... __rcu_free_sheaf_prepare (mm/slub.c:2634 mm/slub.c:2940) rcu_free_sheaf (mm/slub.c:5850) rcu_core (kernel/rcu/tree.c:2617 kernel/rcu/tree.c:2869) handle_softirqs (kernel/softirq.c:622) The buggy address belongs to the cache kmalloc-128 of size 128 Queue the old object for kfree_rcu() only after the new one is published, matching ieee80211_set_probe_resp() and ieee80211_set_s1g_short_beacon().
CVE-2026-64570	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: mac80211: fix fils_discovery double free on alloc failure ieee80211_set_fils_discovery() calls kfree_rcu() on the old template before allocating the replacement. If the kzalloc() then fails, it returns -ENOMEM while link->u.ap.fils_discovery still points at the object already queued for freeing. A later update or AP teardown (ieee80211_stop_ap()) re-queues that same rcu_head; the second free is caught by KASAN when the RCU sheaf is processed in softirq: BUG: KASAN: double-free in rcu_free_sheaf (mm/slub.c:5850) Free of addr ffff88800c065280 by task swapper/0/0 ... __rcu_free_sheaf_prepare (mm/slub.c:2634 mm/slub.c:2940) rcu_free_sheaf (mm/slub.c:5850) rcu_core (kernel/rcu/tree.c:2617 kernel/rcu/tree.c:2869) handle_softirqs (kernel/softirq.c:622) The buggy address belongs to the cache kmalloc-96 of size 96 Queue the old object for kfree_rcu() only after the new one is published, matching ieee80211_set_probe_resp() and ieee80211_set_s1g_short_beacon().

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-64571	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: p54: validate RX frame length in p54_rx_eeprom_readback() p54_rx_eeprom_readback() copies the requested EEPROM slice out of a device-supplied readback frame without checking that the skb actually holds that many bytes. Commit da1b9a55ff11 ("wifi: p54: prevent buffer-overflow in p54_rx_eeprom_readback()") closed the destination overflow by copying a fixed priv->eeprom_slice_size (and rejecting a mismatched advertised len), but the source side is still unbounded: nothing verifies the frame is long enough to supply that many bytes. A malicious USB device can send a short frame whose advertised len matches priv->eeprom_slice_size while the payload is truncated. The equality check passes and memcpy() reads past the end of the skb, leaking adjacent heap: BUG: KASAN: slab-out-of-bounds in p54_rx (drivers/net/wireless/intersil/p54/txrx.c:507) Read of size 1016 at addr ffff88800f077114 by task swapper/0/0 Call Trace: ... __asan_memcpy (mm/kasan/shadow.c:105) p54_rx (drivers/net/wireless/intersil/p54/txrx.c:507) p54u_rx_cb (drivers/net/wireless/intersil/p54/p54usb.c:163) __usb_hcd_giveback_urb (drivers/usb/core/hcd.c:1657) dummy_timer (drivers/usb/gadget/udc/dummy_hcd.c:2005) ... The buggy address belongs to the object at ffff88800f0770c0 which belongs to the cache skbuff_small_head of size 704 The buggy address is located 84 bytes inside of allocated 704-byte region [ffff88800f0770c0, ffff88800f077380) Check that the slice fits in the skb before copying.
CVE-2026-14456	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	openssl 3.5.7	Issue summary: When an OpenSSL QUIC server (Listener SSL object) processes valid QUIC Initial packets for unknown destination connection IDs, it can allocate and queue new incoming channels without enforcing any limit. Impact summary: A remote peer that can make many Initial packets reach the server listener faster than the application accepts connections, can cause the memory allocated to store the per-channel state to grow without any limits, potentially making the QUIC listener unavailable and causing Denial of Service. CWE: CWE-770: Allocation of Resources Without Limits or Throttling Description: The function that handles inbound QUIC packets uses Connection-Id from the packet header to find an existing connection (QUIC channel). If no existing connection is found and the packet type is INITIAL, the function treats the packet as a new connection. It allocates a new channel object and inserts it into a queue where it waits to be accepted by the local application with SSL_accept(3ossl). The memory occupied by these initial channel objects may grow without bounds if the application is not able to call SSL_accept() frequently enough to serve these inbound connection requests. The issue is present since OpenSSL 3.5 when the QUIC server implementation was added. The fix introduces a limit for pending connections. The default limit is set to 256 pending connections (waiting to be accepted by the local application). Applications may change the default by calling SSL_set_value_uint(3ossl). FIPS impact: no The FIPS module is not affected as the QUIC implementation is outside of the OpenSSL FIPS module boundary.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-14457	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	openssl 3.5.7	Issue summary: In a server or client configuration with RFC7250 Raw Public Keys (RPKs) enabled, and only the private key (with no associated certificate) configured locally, a NULL pointer dereference may occur when the remote peer solicits raw public keys and also sends the typically omitted "signature_algorithms_cert" TLS extension. Impact summary: The impact is limited to a possible Denial of Service as a result of an application abort, no data disclosure or remote command execution are possible. CWE: CWE-476: NULL Pointer Dereference Description: While a passing comment in sample code in the documentation suggests that key-only RPK configurations are supported, the best-practice RPK configuration is to always configure a corresponding certificate (possibly self-signed or signed by any convenient CA). When the private key is configured along with a matching certificate, the "signature_algorithms_cert" extension is handled reliably even without the fix, and peer clients or servers that don't support raw public keys may be able to complete a TLS connection by pinning or verifying the corresponding certificate or its public key. Deployments that prefer to configure just a private key with no certificate need to upgrade to an updated release as noted below. FIPS impact: no No FIPS modules are affected by this issue, as the SSL protocol implementation is outside the OpenSSL FIPS module boundary.
CVE-2026-18798	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	openssl 3.5.7	Issue summary: QUIC server may double free QRX (QUIC record layer RX) object when channel creation fails for initial packet. Impact summary: Double free leads to heap corruption, which typically results in termination of QUIC server process, leading to Denial of Service. There is so far no evidence that this double free is exploitable for remote code execution, thus it is considered highly improbable. CWE: CWE-415: Double Free Description: In order to validate initial packet, OpenSSL QUIC stack default packet handler (port_default_packet_handler()) creates a so-called QRX object. If the initial packet validates successfully with QRX object, the default packet handler proceeds to channel (connection object) creation. The QRX object used for packet validation is passed to port_bind_channel(), so it becomes part of the newly created connection. If port_bind_channel() fails, then it also frees the QRX object. Once port_bind_channel() returns, the port_default_packet_handler() detects the failure and proceeds to the error branch, where the same QRX object is freed for the second time. The failure in port_bind_channel() function can be induced with a relatively low effort by a malformed (non RFC 9000 compliant) INITIAL packet. If the packet carries DCID (destination connection ID) which is shorter than 8 bytes, then port_bind_channel() jumps to the error path after ossl_quic_lcidm_enrol_odcid() detects that the DCID has invalid length. FIPS impact: no The FIPS module is not affected, as the QUIC implementation is outside of the OpenSSL FIPS module boundary.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-54874	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	openssl 3.5.7	Issue summary: Receiving a DTLS record for a future epoch while a handshake is in progress causes OpenSSL to buffer far more memory than the record itself requires. Impact summary: A peer can use a small amount of network traffic to make an OpenSSL DTLS endpoint retain a disproportionately large amount of memory, which may lead to a Denial of Service. CWE: CWE-405: Asymmetric Resource Consumption (Amplification) Description: While a DTLS handshake is in progress, a peer may legitimately have already moved on to the next epoch (for example, having sent its ChangeCipherSpec and Finished messages) before the local endpoint has processed the same transition, typically because of reordering on the underlying UDP transport. OpenSSL buffers such early records so that they can be processed once the local endpoint catches up. Buffering a record currently retains the entire read buffer it arrived in, which is sized to hold the largest possible DTLS record (around 16 kilobytes), rather than just the bytes that make up the record itself. Up to 100 such records may be buffered per connection. As a result, a peer that sends a stream of small forged records claiming to belong to the next epoch can cause an OpenSSL DTLS endpoint to retain around 1.7 megabytes of memory, despite sending only a small fraction of that amount of data over the network. An attacker therefore gains a memory amplification factor of around 1200, and can multiply the effect across as many associations as it is able to open, making this a remote memory exhaustion Denial of Service risk for DTLS servers. Since the memory retained per connection remains bounded, and any limit an application already places on the number of concurrent associations also bounds the total exposure, this issue has been assessed as Low severity. FIPS impact: no No FIPS modules are affected by this issue as the affected code is outside the OpenSSL FIPS module boundary. OpenSSL 4.0, 3.6, 3.5, 3.4, 3.0, 1.1.1 and 1.0.2 are vulnerable to this issue. OpenSSL 4.0 users should upgrade to OpenSSL 4.0.2. OpenSSL 3.6 users should upgrade to OpenSSL 3.6.4. OpenSSL 3.5 users should upgrade to OpenSSL 3.5.8. OpenSSL 3.4 users should upgrade to OpenSSL 3.4.7. OpenSSL 3.0 users should upgrade to OpenSSL 3.0.22. Premium support customers only: OpenSSL 1.1.1 users should upgrade to OpenSSL 1.1.1zi OpenSSL 1.0.2 users should upgrade to OpenSSL 1.0.2zr This issue was reported on 18 May 2026 by Amazon Web Services. The fix has been developed by Matt Caswell. -- cut (non-publishing metadata for internal use) -- Reported by: Amazon Web Services Fixed by: Matt Caswell

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-63072	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	openssl 3.5.7	Issue summary: OpenSSL CMS decryption sizes the key-unwrap output buffer based on querying the unwrapped key size, but the AES-WRAP-PAD unwrap primitive can write and cleanse more bytes than that query reports, causing an 8-byte out-of-bounds heap write. Impact summary: An attacker who supplies a crafted CMS message can trigger a deterministic 8-byte out-of-bounds heap write when the victim decrypts it with CMS_decrypt(), corrupting the heap and typically resulting in a Denial of Service. CWE: CWE-787: Out-of-bounds Write Description: The key-wrap OID is potentially attacker-controlled on the wire. CMS unwrapping allows both id-aesNNN-wrap-pad and id-aesNNN-wrap ciphers. An attacker can take a legitimate message and change a single OID byte to select the padded variant while leaving the message otherwise valid. Since the unwrap key is derived from the recipient's private operation (ECDH key agreement or ML-KEM decapsulation), the RFC 5649 integrity check cannot pass, and the decryption fails with integrity failure. The write is a fixed-size (8-byte), fixed-value (zero) heap overflow immediately past the allocation, requires no special configuration, and is reachable from the public CMS_decrypt() function. The consequence is a heap corruption leading to a Denial of Service. The fix in the CMS code sizes the unwrap output buffer for the worst case so a failed unwrap cannot write past the allocation. FIPS impact: no As the CMS code lives outside the FIPS module boundary, no FIPS modules are affected by this CVE.
CVE-2026-63075	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	openssl 3.5.7	Issue summary: When OpenSSL processes QUIC traffic from a peer that repeatedly sends ack-eliciting packets while not acknowledging ACK-only responses, the QUIC stack can retain ACK-only packet metadata for the lifetime of the connection. Impact summary: A remote peer that can complete a QUIC handshake can cause connection-scoped memory growth which may lead to Denial of Service through memory exhaustion, especially with sustained traffic or many concurrent QUIC connections. CWE: CWE-770: Allocation of Resources Without Limits or Throttling Description: When the OpenSSL QUIC stack sends an ACK-only packet, there is no requirement by the QUIC protocol that the peer will acknowledge that ACK-only packet (i.e. it is itself not ack-eliciting). However, the OpenSSL implementation stores the metadata about the ACK frames regardless. In and of itself that's ok, but if a malicious peer establishes a connection, and then drives the connection such that ACK-only packets are forced from the OpenSSL implementation peer (i.e., by sending numerous PING frames), and then withholding any subsequent acks for ack-eliciting data, like legitimate data, said malicious peer can force inappropriate memory growth on the OpenSSL peer, potentially leading to a Denial of Service. The fix is to ensure that we account for the transmission of the ACK-only packet in the packet histories high and low watermark without actually storing the ACK-only packet metadata itself. FIPS impact: no The OpenSSL FIPS module is not affected as the QUIC code is outside the FIPS module boundary.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-63076	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	openssl 3.5.7	Issue summary: OpenSSL CMP password based protection verification only checks whether the protectionAlg parameter was not NULL and not its ASN.1 type, before treating it as a PBMParameter. A crafted message can contain a parameter of a different type, which is then dereferenced as an invalid pointer. Impact summary: A remote, unauthenticated attacker can crash an application acting as a CMP server that accepts PBM-protected messages, or a CMP client talking to a malicious or intercepted CMP server, resulting in a Denial of Service. CWE: CWE-476: NULL Pointer Dereference Description: When verifying the password-based MAC protection of a CMP message, OpenSSL library reads the protectionAlg algorithm parameter with X509_ALGOR_get0(), which returns both the parameter type and its value pointer. The value is then cast to an ASN1_STRING and treated as the expected PBMParameter after only checking that pointer is not NULL. The parameter type returned by X509_ALGOR_get0() was never consulted. This happens during protection verification, before any MAC is computed, so no knowledge of the PBM shared secret is required; the only precondition is that PBM verification is reachable. On the server side this is reached from OSSL_CMP_SRV_process_request() for any application that stands up a CMP server accepting PBM-protected messages, and on the client side from CMP response validation against a malicious or on-path (MITM) server. The reliable consequence is a denial of service; there is no memory disclosure, no controlled memory write, and no path to code execution. CMP is a specialized feature that an application must explicitly enable. FIPS impact: no As the CMP code lives outside the FIPS module boundary, no FIPS modules are affected by this CVE.
CVE-2026-13608	HIGH	7.4	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N	curl 8.7.1	A flaw in the libcurl SASL negotiation for LDAP authentication allows an incomplete handshake sequence to be misinterpreted as a successful cryptographic verification. An attacker executing a Man-in-the-Middle (MITM) attack can inject a premature or shortcut response that bypasses complete peer validation.
CVE-2026-82209	HIGH	8.2	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:H/A:N	curl 8.7.1	When libpsl support is enabled, libcurl fails to enforce the Public Suffix List boundary check when processing a `Set-Cookie` header where the `Domain` attribute explicitly matches an origin host that is itself a public suffix (e.g., `Domain=co.uk` set by `co.uk`). Instead of coercing it into a strict host-only cookie, libcurl saves the cookie with wildcard domain scope (`.co.uk`). Consequently, the cookie is inappropriately included in subsequent outbound requests or HTTP redirects to arbitrary sibling subdomains under the same public suffix (e.g., `attacker.co.uk`).
CVE-2026-98052	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: bcmasp: clear txcb->last before writing each descriptor bcmasp_xmit() only wrote txcb->last = true for the final fragment of an SKB; non-final fragments left the field untouched. If a descriptor slot was reused while it still held a stale true from a previous SKB (possible when tx_spb_ring_full() underreported fullness), bcmasp_tx_reclaim() would see last == true mid-SKB and call dev_consume_skb_any() prematurely, freeing the sk_buff while its remaining fragments were still in flight. Unconditionally clear txcb->last before the conditional set so every descriptor slot starts from a known false state regardless of what a prior transmission left behind.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98056	HIGH	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: nvme: remove stale namespaces by NSID range during scan nvme_scan_ns_list() drops the stale namespaces in each gap in the reported NSID list one NSID at a time. Every iteration calls nvme_find_get_ns() to look the namespace up and removes it if it is present. The loop runs once per NSID in the gap rather than once per namespace actually present. NSIDs are 32-bit, so a target with a sparse NSID space can make a single gap spin the loop billions of times with nothing to remove. watchdog: BUG: soft lockup - CPU#4 stuck for 26s! Workqueue: nvme-wq nvme_scan_work [nvme_core] RIP: 0010:__srcu_read_unlock+0xb/0x20 Call Trace: nvme_find_get_ns+0x7d/0xb0 [nvme_core] nvme_scan_ns_list+0xe8/0x280 [nvme_core] nvme_scan_work+0x18a/0x280 [nvme_core] process_one_work+0x197/0x380 worker_thread+0x2fe/0x410 kthread+0xe0/0x100 Rename nvme_remove_invalid_namespaces() to nvme_remove_nsid_range() and give it an open (start, end) NSID range. ctrl->namespaces is sorted by NSID, so the whole gap is dropped in a single walk that stops once end is reached. This bounds the work by the namespaces that are present instead of by the size of the gap.
CVE-2026-98066	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ALSA: caiaq: Fix potential double-free at error path The fix for caiaq driver's resource management to handle the errors tries to release the resources in a common destructor call, but as a sashiko review for another patch suggested, some of the audio resources such as URBs have been already freed, and this may lead to a double-free. For addressing the double-free, call the common destructor function from each place, and assure that the resource pointers get cleared.
CVE-2026-98069	HIGH	8.1	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net/rds: acquire the fastpath locks in rds_conn_shutdown() rds_conn_shutdown() quiesces the transmit and receive-refill paths by waiting for RDS_IN_XMIT and RDS_RECV_REFILL to be sampled clear, and then runs the transport shutdown and rds_conn_path_reset(). Sampling the bits clear is not the same as owning them: the moment after the wait_event() returns, rds_send_xmit() can re-acquire RDS_IN_XMIT (or rds_ib_recv_refill() can re-acquire RDS_RECV_REFILL) and run concurrently with the teardown. The sender does recheck the connection state after taking the lock, but that recheck is a classic store-buffering pattern: teardown writes the state and reads the bit while the sender writes the bit and reads the state. acquire_in_xmit() is only an acquire operation, so on weakly ordered architectures both sides can miss each other's write, and the transmit path then runs while the transport zeroes its rings (e.g. rds_ib_ring_init()) and rds_send_path_reset() rewrites the transmit state under it. Oracle UEK fixed the same class of crashes - a 14-year tail of BUG_ON()s in rds_ib_sub_signaled(), unexpected op-codes and NULL dereferences in rds_ib_send_cqe_handler() during failover testing - by making the teardown path *acquire* the fastpath bit locks instead of testing them ("rds: Make sure transmit path and connection tear-down does not run concurrently"). Ownership of a single word is decided by RMW atomicity, so no cross-variable ordering is needed. Do the same here: take both locks before calling the transport shutdown, hold them across rds_conn_path_reset(), and release them explicitly with a wake-up afterwards. Both are released with clear_bit_unlock(), so that the ring re-initialization done by the transport shutdown and the transmit state rewritten by

CVE	Severity	CVSS	CVSS Vector	Component	Description
					rds_send_path_reset() are ordered before either bit is seen clear by the next acquire_in_xmit() or acquire_refill(). The fastpath users of these bits - rds_send_xmit() and rds_ib_recv_refill() - are trylock style and back off while teardown owns the locks, so no new lock dependency is introduced for them. rds_tcp_reset_callbacks() is different: since the previous patch it acquires RDS_IN_XMIT as well, and it blocks doing so, so its wait now spans the teardown instead of at most one send batch. That waiter runs from rds_tcp_accept_one() on the single-threaded krdsd workqueue and holds rds_tcp_accept_lock and t_conn_path_lock while it waits, so a duelling SYN accepted while its path is being torn down parks accept processing for the duration of the teardown - for TCP bounded by the (up to 5 s) drain loop in rds_tcp_conn_path_shutdown(). An IB path's drain in rds_ib_conn_path_shutdown() has no round cap, but no blocking waiter either: rds_tcp_reset_callbacks() is the only blocking acquirer of these bits and waits only on its own TCP path, and the fastpaths are trylock-and-back-off on both transports, so a long IB drain lengthens only that path's own quiesce. The window is narrow: the accept-side state check has to pass before the teardown moves the path to RDS_CONN_DISCONNECTING. Because krdsd is a single global workqueue, everything else queued there - accept processing for other connections and network namespaces, and the flush_workqueue(rds_wq) in rds_tcp_listen_stop() during namespace teardown - waits behind the parked accept worker for that time. It cannot deadlock, although the waits do point at each other: the teardown blocks until the bit's holder releases it, and the holder may be that krdsd accept worker. The holder finishes without needing anything the teardown owns: the sync cancels rds_tcp_reset_callbacks() issues target cp_send_w and cp_recv_w on the path's ordered cp_wq, whose only execution slot is occupied by the blocked cp_down_w itself, so they are pending at most and cancel without flushing - a reliance on cp_wq being ordered that is now noted next to those cancels (on ---truncated---

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98070	HIGH	8.1	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net/rds: acquire RDS_IN_XMIT in rds_tcp_reset_callbacks() rds_tcp_reset_callbacks() quiesces the transmit path by setting the path state to RDS_CONN_RESETTING and then waiting for RDS_IN_XMIT to be sampled clear before swapping the underlying socket and calling rds_send_path_reset(). Sampling the bit clear is not the same as owning it: rds_send_xmit() can re-acquire RDS_IN_XMIT right after the wait_event() returns. Its state recheck after taking the lock is a store-buffering pattern (the resetter writes the state and reads the bit, the sender writes the bit and reads the state) and acquire_in_xmit() is only an acquire operation, so on weakly ordered architectures both sides can miss each other's write and the transmit path then runs concurrently with rds_send_path_reset() rewriting cp_xmit_* state - which is exactly what the comment above rds_send_path_reset() tells its callers to prevent. Take the lock instead, hold it across the socket swap and rds_send_path_reset(), and release it with a wake-up at the end. The lock-ordering constraint documented above the wait still holds: the lock is acquired before lock_sock(), so a sender inside tcp_sendmsg() can never be waited on while we hold the socket lock. Two details of the old code go away with the same change: - t_sock is now read only after the lock is acquired. The old code cached it before waiting; the teardown in rds_conn_shutdown() releases that socket and clears t_sock, so a pointer cached before the wait can be stale by the time the accept path resumes. Reading it under RDS_IN_XMIT is what makes the exclusion complete once the teardown owns the same lock, which the next patch arranges; until then the teardown still only samples the bit, and the two paths remain as exposed to each other as they are today. - The old !osock early path called rds_send_path_reset() with no serialization at all. It now runs under the lock like the normal path. The conditional RDS_CONN_RESETTING transition of the previous patch happens before the socket check either way: a path found without a socket is either still connecting (its reconnect worker blocked on t_conn_path_lock) and legitimately goes RESETTING -> UP on the new socket, or it has been torn down meanwhile and is dropped. The in-function comment describing the old wait-based quiesce is rewritten to describe the lock-based one, and the stale block comment above the function (which still described a return value and an incomplete list of t_sock writers) is refreshed to name all four writers - the connect, accept, teardown and swap paths - and what serializes each of them.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98071	HIGH	7.0	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net/rds: clear cp_flags bits individually in rds_conn_path_reset() rds_conn_path_reset() wipes the whole flag word with a plain cp->cp_flags = 0 store. Every other accessor of that word uses atomic bitops, and some of them can run concurrently with the reset: RDS_LL_SEND_FULL is set from rds_send_xmit() and cleared from the transport completion paths, neither of which holds anything that excludes the shutdown worker. A plain store racing an atomic read-modify-write on the same word is a data race, and whichever side loses has its update silently discarded. Clear the two bits the reset is actually responsible for instead. RDS_IN_XMIT and RDS_RECV_REFILL need no store at all here: they belong to the caller, rds_conn_shutdown(), which waits for both to be clear before calling the transport shutdown and this reset. This also gives every bit in cp_flags a single well-defined writer discipline, which the following patches rely on when they turn RDS_IN_XMIT and RDS_RECV_REFILL into bit locks held across the teardown: a blanket store mid-teardown would destroy lock ownership that an atomic clear preserves. Oracle UEK carries the same conversion ("net/rds: Preserve essential connection state flags"), motivated by its asynchronous shutdown state machine, whose progress and destroy flags must survive the reset. UEK's variant also clears RDS_IN_XMIT and RDS_RECV_REFILL because there the reset runs as the final step of a teardown that owns both bits, making those clears its unlock. Upstream that release belongs in rds_conn_shutdown(): once a later patch in this series turns the two bits into locks held across the teardown, ending ownership needs release semantics and a wake-up that a plain clear inside the reset would not provide. Based on Oracle UEK commit "net/rds: Preserve essential connection state flags" by Gerd Rausch.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98073	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: Remove conflicting altnames for dying netns in __dev_change_net_namespace(). syzbot reported the warning in cfg80211_pernet_exit(). [0] The repro does the following: 1. create two device in root netns and non-root netns 2. assign the same altname for the two devices 3. remove the non-root netns Since commit 7663d522099e ("net: check for altname conflicts when changing netdev's netns"), cfg80211_switch_netns() and cfg802154_switch_netns() fail if init_net has a device with the conflicting altname. default_device_exit_net() had the same issue and commit d09486a04f5d ("net: fix removing a namespace with conflicting altnames") fixed it. cfg80211_pernet_exit() and cfg802154_pernet_exit() need the same fix. Let's generalise the fix by removing conflicting altnames for dying netns in __dev_change_net_namespace(). [0]: cfg80211_switch_netns(rdev, &init_net) WARNING: net/wireless/core.c:1871 at cfg80211_pernet_exit+0xd5/0x120 net/wireless/core.c:1871, CPU#1: kworker/u8:9/1160 Modules linked in: CPU: 1 UID: 0 PID: 1160 Comm: kworker/u8:9 Not tainted syzkaller #0 PREEMPT(full) Hardware name: Google Google Compute Engine/Google Compute Engine, BIOS Google 07/24/2026 Workqueue: netns cleanup_net RIP: 0010:cfg80211_pernet_exit+0xd5/0x120 net/wireless/core.c:1871 Code: e8 03 42 80 3c 20 00 74 08 4c 89 f7 e8 b4 ef 0e f7 4d 8b 36 49 81 fe 20 10 4a 90 74 12 e8 03 3d 9f f6 eb 85 e8 fc 3c 9f f6 90 <0f> 0b 90 eb cc e8 f1 3c 9f f6 eb 05 e8 ea 3c 9f f6 5b 41 5c 41 5e RSP: 0018:ffffc900057a78f0 EFLAGS: 00010293 RAX: ffffffff8b287154 RBX: ffff88807ba72780 RCX: ffff8880213e8000 RDX: 0000000000000000 RSI: 00000000ffffffef RDI: 0000000000000000 RBP: 00000000ffffffef R08: ffffffff9024cc67 R09: 0000000000000000 R10: fffff52000af4eb0 R11: fffffbfff204998d R12: dffffc0000000000 R13: ffffffff904a1080 R14: ffff888144ed0008 R15: ffff888144ed0e20 FS: 0000000000000000(0000) GS:ffff888124de6000(0000) knlGS:0000000000000000 CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033 CR2: 00005642de0a8a70 CR3: 000000007a40c000 CR4: 00000000003526f0 Call Trace: ops_exit_list net/core/net_namespace.c:200 [inline] ops_undo_list+0x43d/0x8d0 net/core/net_namespace.c:253 cleanup_net+0x572/0x810 net/core/net_namespace.c:706 process_one_work kernel/workqueue.c:3387 [inline] process_scheduled_works+0xc3d/0x1630 kernel/workqueue.c:3470 worker_thread+0xa47/0xfb0 kernel/workqueue.c:3551 kthread+0x38b/0x480 kernel/kthread.c:436 ret_from_fork+0x514/0xb70 arch/x86/kernel/process.c:158 ret_from_fork_asm+0x1a/0x30 arch/x86/entry/entry_64.S:245

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98076	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: tracing/probes: Fix use-after-free on field name/type of events with multiple probes The fields of a probe-based dynamic event (kprobe, uprobe, eprobe and fprobe events) are created in traceprobe_define_arg_fields() by handing the probe_arg name/type strings to trace_define_field(), which only stores the pointers without copying. Those strings are owned by the trace_probe and are freed when that probe is removed. An event can have several probes attached. The field list is defined only once, by the first probe that registers the event, but it is kept alive by any surviving sibling probe. Deleting just that first probe by symbol - # primary A: fields are defined from A's args echo 'p:kprobes/ev vfs_read a1=$arg1' > kprobe_events # append B: shares A's event call echo 'p:kprobes/ev vfs_write a1=$arg1' >> kprobe_events # delete only A (matched by symbol), B survives echo '-:kprobes/ev vfs_read' >> kprobe_events frees A's args (trace_probe_cleanup() -> traceprobe_free_probe_arg()), but trace_probe_unlink() keeps the trace_probe_event because the probe list is not empty. The event call stays registered via B while its fields now reference freed memory. Any field lookup then reads it, e.g. echo 'a1 == 1' > events/kprobes/ev/filter BUG: KASAN: slab-use-after-free in strcmp+0xa7/0xb0 Call Trace: strcmp trace_find_event_field parse_pred process_preds create_filter apply_event_filter event_filter_write field->name references parg->name (kstrdup'd, freed with the probe) and, for array arguments, field->type references parg->fmt (kmalloc'd, freed with the probe) - the scalar type otherwise points at the static fmttype rodata, which is safe. Have traceprobe_define_arg_fields() duplicate the name and type strings and anchor the copies on the trace_probe_event, which embeds the event call and outlives every individual probe; trace_probe_event_free() releases them. The reproducer above triggers reliably; the field lookup and the delete both run under event_mutex, so this is a dangling reference after removal rather than a race. The issue was found by the autokbug dynamic kernel fuzzer at Tencent Yunding Lab.
CVE-2026-98077	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: nf_conntrack_sip: fix OOB read in sip_skip_whitespace() sip_skip_whitespace() returns dptr unchanged when its own loop exhausts the buffer (dptr == limit), instead of NULL like its sibling sip_follow_continuation() returns on its own "no more data" path. ct_sip_get_header() only checks for NULL after calling it: dptr = sip_skip_whitespace(dptr, limit); if (dptr == NULL) break; if (*dptr != ':' || ++dptr >= limit) break; so a recognized header name followed only by spaces/tabs running to the exact end of the SIP payload, with no colon, makes the very next statement read one byte past the buffer. Make both "no more data" outcomes return NULL, matching the convention sip_follow_continuation() already uses and that both existing callers already check for.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98083	HIGH	7.0	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: btrfs: fix transaction use-after-free in raid stripe insertion If allocation of a RAID stripe extent fails, btrfs_insert_one_raid_extent() aborts and ends the transaction before returning -ENOMEM. btrfs_finish_one_ordered(), the production caller through btrfs_insert_raid_extent(), still owns the transaction handle. It handles the error by aborting the transaction and then reaches the common exit path, which ends the transaction again. The premature end can free the handle and drop its transaction reference. Transaction cleanup can then free the transaction before the caller's second abort accesses the handle and transaction, resulting in use-after-free. Keep the abort at the failure site, but let the caller's common exit path end the transaction once, after it has finished using both objects.
CVE-2026-98088	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: scsi: mpt3sas: Avoid out-of-bounds cpumask_of_node() call in _base_assign_reply_queues() dev_to_node() can return NUMA_NO_NODE (-1) on systems without NUMA topology information for the PCI device, such as single-socket boards that don't expose device-to-node affinity. Passing -1 directly into cpumask_of_node() indexes node_to_cpumask_map[-1], an out-of-bounds array read caught by UBSAN: UBSAN: array-index-out-of-bounds in arch/x86/include/asm/topology.h:72:28 index -1 is out of range for type 'cpumask *[1024]' Fall back to cpu_online_mask when no NUMA node is available, rather than assuming dev_to_node() always returns a valid node index.
CVE-2026-98096	HIGH	7.4	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ipv6: sr: restore network header before routing and forwarding ipv6_srh_rcv() runs with skb->data at the Segment Routing Header (SRH) while skb_network_header() points at the IPv6 header. When segments_left > 0, ipv6_srh_rcv() previously restored the skb->data position by pushing sizeof(struct ipv6hdr), assuming the SRH immediately followed the fixed IPv6 header. If another extension header (such as a Hop-by-Hop options header) precedes the SRH, skb_network_offset() remained negative. This led to two problems: 1. During ip6_route_input(), fib6_rules_early_flow_dissect() invokes __skb_flow_dissect() which passes the negative skb_network_offset() to flow dissection, breaking BPF and C flow dissector logic. 2. If forwarded via ip6_forward() or redirected via act_mirred, downstream handlers (like sch_fragment() or neighbour output) pass the negative offset as an unsigned length, triggering OOB memcpy or buffer overflows. Fix this by pushing -skb_network_offset(skb) before routing, ensuring skb_network_offset(skb) is 0 for route lookup / flow dissection as well as downstream forwarding. On the loopback path, pull skb_transport_offset(skb) to restore skb->data to the SRH before looping back.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98103	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: igmp: convert struct ip_sf_list to RCU Commit 23d2b94043ca ("igmp: Add ip_mc_list lock in ip_check_mc_rcu") added spin_lock_bh(&im->lock) to ip_check_mc_rcu() to prevent a use-after-free while iterating im->sources during concurrent deletions. However, ip_check_mc_rcu() is called from RCU read-side critical sections in packet receive and route lookup fast paths (e.g. __mkroute_output(), ip_route_input_rcu(), and __udp4_lib_rcv()). When igmpv3_send_cr() or igmpv3_send_report() holds &pmc->lock and calls add_grec() -> igmpv3_newpack() -> ip_route_output_ports(), an XFRM policy matching a multicast destination triggers xfrm_tmpl_resolve_one() -> xfrm4_get_saddr() -> __mkroute_output() -> ip_check_mc_rcu(). This attempts to acquire &im->lock while &pmc->lock is already held on the same CPU, triggering a lockdep recursive locking warning / deadlock. Fix this by converting IPv4 struct ip_sf_list to RCU, mirroring the IPv6 implementation in net/ipv6/mcast.c: 1. Add struct rcu_head to struct ip_sf_list and annotate sf_next, sources, and tomb as __rcu pointers. 2. Use rcu_assign_pointer() and kfree_rcu() for list updates and deletions. 3. Remove spin_lock_bh(&im->lock) from ip_check_mc_rcu() and traverse im->sources locklessly with for_each_psf_rcu(), reading and writing counter fields with READ_ONCE() and WRITE_ONCE(). Note: RCU conversion of /proc/net/mcfilter will be done in a separate patch.
CVE-2026-98107	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: Bluetooth: L2CAP: fix out-of-bounds write in l2cap_ecred_connect l2cap_chan_connect() tries to ensure there are no more than L2CAP_ECRED_CONN_SCID_MAX pending ECRED channels, so they fit in the same L2CAP_ECRED_CONN_REQ that l2cap_ecred_connect() constructs. However, the check only counts deferred channels. If 6 L2CAP sockets are connected at the same time in order DDDDND (D=deferred, N=non-deferred), the last can bump the total to max+1. It results to one __le16 written out of bounds of the scid array, and an invalid ECRED_CONN_REQ being sent. Fix by leaving room for the non-deferred pending ECRED channels in the counting in l2cap_chan_connect(), so the limit can't be exceeded. Move counting under same critical section where the channel is added. Although race conditions involving this appear unreachable, it's easier to see. Also add WARN_ON_ONCE check in l2cap_ecred_defer_connect() to make this less brittle.
CVE-2026-98108	HIGH	7.5	CVSS:3.1/AV:A/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: Bluetooth: L2CAP: fix chan mode for LE_CONN_REQ + EXT_FLOWCTL pchan l2cap_new_connection() sets default value of channel mode to match the parent channel. l2cap_le_connect_req() left this at the default, and created L2CAP_MODE_EXT_FLOWCTL channels if listening pchan has that mode. This causes FLAG_DEFER_SETUP channels to reply to L2CAP_LE_CONN_REQ with L2CAP_ECRED_CONN_RSP, which is incorrect. It can also result to stack OOB write (of l2cap_alloc_cid determined values) in l2cap_ecred_rsp_defer(), as l2cap_le_connect_req() does not limit maximum number of deferred channels or check for duplicate ident. Fix by setting chan->mode correctly in l2cap_le_connect_req(). Also check channel mode in l2cap_ecred_rsp_defer(), and do WARN_ON_ONCE instead of OOB write to make it less brittle.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98110	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: Bluetooth: btintel: bound firmware ID by TLV length The firmware ID is treated as a NUL-terminated string even though the TLV length is its only boundary. If the value does not contain a NUL terminator, snprintf() can read beyond the received response. Limit the conversion to the advertised TLV value length.
CVE-2026-98111	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: Bluetooth: btintel: validate version TLV value lengths btintel_parse_version_tlv() verifies that a complete TLV is present in the response, but it does not ensure that the value is long enough for the specific TLV type. A short value can therefore cause an out-of-bounds read through get_unaligned_le16(), get_unaligned_le32(), or memcpy(). Reject values shorter than the minimum required by each known TLV type. Also reject responses that do not contain the Command Complete Status field.
CVE-2026-98116	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ALSA: pcm: Serialize PCM mmap with buffer reallocation to fix page UAF snd_pcm_hw_params() and snd_pcm_hw_free() guard buffer reallocation with an mmap_count check performed under the PCM stream lock, but the lock is released long before the buffer is actually freed: snd_pcm_sync_stop(), constraint refinement and do_free_pages() all happen in between. snd_pcm_mmap_data(), on the other hand, takes no lock at all: it validates against the old buffer's state and dma_bytes, remaps its pages into the VMA, and only then increments mmap_count. A concurrent mmap() can therefore slip in between the check and the free. remap_pfn_range() installs writable PTEs for the old buffer's pages without taking page references, and the subsequent do_free_pages() returns those pages to the page allocator while the VMA still maps them. This leaves a stale, writable mapping of freed pages: a page-level use-after-free that can be leveraged for local privilege escalation. Make snd_pcm_mmap_data() participate in the buffer-access scheme introduced for hw_params/hw_free: acquire runtime->buffer_accessing before validating and remapping, and release it afterwards. Buffer reallocation already fails with -EBUSY while accessors are active, and the mmap side now fails with -EBUSY while a reallocation is in progress, so the validate/remap sequence and the check/free sequence can no longer interleave. A reproducer that turns this race into a stale writable mapping of the freed DMA buffer pages is available on request.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98122	HIGH	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: vxlan: mdb: Fix use-after-free in vxlan_mdb_remote_src_del() vxlan_mdb_is_valid_source(), which validates MDBE_ATTR_SOURCE and every MDBE_ATTR_SRC_LIST member, accepts the all-zeros address. A source list is only accepted on a (*, G) entry, whose source is the all-zeros address, and for each member of the list an (S, G) entry is derived from it by substituting the source. Entries are keyed by a plain memcmp() of struct vxlan_mdb_entry_key, so if MDBE_ATTR_SOURCE is present and holds the all-zeros address and the source list holds it as well, the derived (S, G) key is byte-identical to the (*, G) key and resolves to the same entry. Omitting MDBE_ATTR_SOURCE is not equivalent, as the key is then left with a zero address family. vxlan_mdb_remote_src_del() removes the forwarding entry of a source before freeing the source entry: vxlan_mdb_remote_src_fwd_del(vxlan, group, remote, &ent->addr); vxlan_mdb_remote_src_entry_del(ent); With the keys aliased, the first call deletes the remote of the entry that owns 'ent' instead of a separate (S, G) entry, and frees 'ent'. The second call then runs on the freed entry, and its hlist_del() reads ->pprev and ->next out of it and writes through them. Adding the (*, G) entry with NLM_F_REPLACE and no source list marks the all-zeros source for deletion and reaches this from the sweep at the end of vxlan_mdb_remote_srcs_replace(). BUG: KASAN: slab-use-after-free in __vxlan_mdb_add+0x1cd/0xd70 Read of size 8 at addr ffff888102852500 by task poc/84 __vxlan_mdb_add+0x1cd/0xd70 vxlan_mdb_add+0xc0/0x140 rtnl_mdb_add+0x157/0x2a0 rtnetlink_rcv_msg+0x207/0x5a0 Allocated by task 84: __kmalloc_cache_noprof+0x153/0x360 vxlan_mdb_remote_srcs_add+0x2eb/0x440 __vxlan_mdb_add+0x803/0xd70 Freed by task 84: kfree+0x14c/0x3b0 vxlan_mdb_remote_del+0x129/0x1a0 __vxlan_mdb_del+0x4f/0xe0 vxlan_mdb_remote_src_fwd_del.isra.0+0x162/0x1b0 __vxlan_mdb_add+0x1c5/0xd70 The MDB operations are netns-scoped, so an unprivileged user can perform them in a new user and network namespace. Reject the all-zeros address in vxlan_mdb_is_valid_source(), which covers both call sites. A (*, G) entry is expressed by omitting the source, so nothing legitimate is refused. Discovered by XBOW, triaged by Baul Lee
CVE-2026-98130	HIGH	8.1	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: sctp: fix a TOCTOU race in SCTP_CMD_TIMER_START The SCTP_CMD_TIMER_START handler checks timer_pending() before calling timer_reduce(). The timer can expire and detach between these operations, causing timer_reduce() to rearm the timer without taking the association reference required for the newly armed timer. The timer callback later unconditionally drops its association reference, which can leave the association reference count unbalanced and result in use-after-free during association teardown. Use the return value of timer_reduce() to determine whether the timer was actually armed. Take the association reference only when timer_reduce() successfully starts a new timer, closing the race between checking the timer state and rearming it. This issue was reported by Nico Yip (@_cyeaa_) working with TrendAI Zero Day Initiative.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98154	HIGH	7.0	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: nvme-rdma: fix -EIO cleanup order in queue_rq On -EIO, the RDMA queue_rq path reports a host path error and then still cleans up the command and unmaps the SQE DMA. The path error helper completes the request, so that is double cleanup and DMA unmap after the request is already complete. Unmap the SQE first, then report the host path error. Skip the outer command cleanup on that path.
CVE-2026-98155	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: accel/qaic: Address potential out-of-bounds read in resp_worker() Although 'commit 2feec5ae5df7 ("accel/qaic: Handle DBC deactivation if the owner went away")' fixes the scenario it was intended for by walking the message and only decoding QAIC_TRANS_DEACTIVATE_FROM_DEV, if present, it skipped over the bounds checking code that is included in decode_message(). This could lead to issues such as reading past the slab allocation's end, infinite loops or kernel panics. For those issues to happen, a malformed wire message is needed to be sent from the device. Instead of duplicating the bounds checking code already present in decode_message(), use the function inside resp_worker().
CVE-2026-98156	HIGH	7.8	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/virtio: use the DMA API for resource backing on Xen On a Xen PV domain page addresses bear no relation to the real machine addresses the host would have to use to reach it. virtio_ring.c handles this correctly, vring_use_map_api() returns true for any xen_domain() regardless of VIRTIO_F_ACCESS_PLATFORM. virtio-gpu makes the same decision independently, but its copy looks only at the feature bit: bool use_dma_api = !virtio_has_dma_quirk(vgdev->vdev); QEMU does not set iommu_platform on virtio-vga by default, so VIRTIO_F_ACCESS_PLATFORM is not negotiated, use_dma_api is false, and virtio_gpu_object_shmem_init() describes the framebuffer's backing pages to the host with sg_phys(). Those are guest-physical addresses. In a PV domain they resolve, on the host side, to pages belonging to some other domain, so the host scans out unrelated memory. Move the decision into virtio_gpu_use_dma_api() and give it the xen_domain() check, like vring_use_map_api() has. This additionally enables the dma_sync_sgtable_for_device() calls in virtgpu_vq.c, which are required for correctness whenever swiotlb is in play. Reproduced with a Xen 4.21 PV dom0 nested inside QEMU 8.2 with virtio-vga, on both a distro 6.8 kernel and 6.18 LTS. A PVH dom0 works fine and doesn't need this fix because it is identity-mapped, only PV dom0s are affected.
CVE-2026-98164	HIGH	7.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: KVM: x86/mmu: Check write tracking in all address spaces kvm_gfn_is_write_tracked() checks only the supplied memslot, but page tracking is per-address-space and shadow pages are shared across all address spaces. With SMM, a GFN can therefore be write-tracked in one address space and appear untracked through the other. Check the supplied slot first, then the slot for the other address space. This ensures all callers honor write tracking regardless of the active address space. In particular, it prevents mmu_try_to_unsync_pages() from marking an upper-level shadow page unsync and eventually triggering the BUG in pte_list_remove(). [invert direction of the conditional. - Paolo]

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2022-4543	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N	linux-rittal 6.12.100+git	A flaw named "EntryBleed" was found in the Linux Kernel Page Table Isolation (KPTI). This issue could allow a local attacker to leak KASLR base via prefetch side-channels based on TLB timing for Intel systems.
CVE-2021-3669	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	A flaw was found in the Linux kernel. Measuring usage of the shared memory does not scale with large shared memory segment counts which could lead to resource exhaustion and DoS.
CVE-2011-1548	MEDIUM	N/A	N/A	logrotate 3.21.0	The default configuration of logrotate on Debian GNU/Linux uses root privileges to process files in directories that permit non-root write access, which allows local users to conduct symlink and hard link attacks by leveraging logrotate's lack of support for untrusted directories, as demonstrated by /var/log/postgresql/.
CVE-2011-1549	MEDIUM	N/A	N/A	logrotate 3.21.0	The default configuration of logrotate on Gentoo Linux uses root privileges to process files in directories that permit non-root write access, which allows local users to conduct symlink and hard link attacks by leveraging logrotate's lack of support for untrusted directories, as demonstrated by directories under /var/log/ for packages.
CVE-2011-1550	MEDIUM	N/A	N/A	logrotate 3.21.0	The default configuration of logrotate on SUSE openSUSE Factory uses root privileges to process files in directories that permit non-root write access, which allows local users to conduct symlink and hard link attacks by leveraging logrotate's lack of support for untrusted directories, as demonstrated by directories for the (1) cobbler, (2) inn, (3) safte-monitor, and (4) uucp packages.
CVE-2010-4563	MEDIUM	N/A	N/A	linux-rittal 6.12.100+git	The Linux kernel, when using IPv6, allows remote attackers to determine whether a host is sniffing the network by sending an ICMPv6 Echo Request to a multicast address and determining whether an Echo Reply is sent, as demonstrated by thcping.
CVE-2010-4756	MEDIUM	N/A	N/A	glibc 2.39+git	The glob implementation in the GNU C Library (aka glibc or libc6) allows remote authenticated users to cause a denial of service (CPU and memory consumption) via crafted glob expressions that do not match any pathnames, as demonstrated by glob expressions in STAT commands to an FTP daemon, a different vulnerability than CVE-2010-2632.
CVE-2007-2768	MEDIUM	N/A	N/A	openssh 9.6p1	OpenSSH, when using OPIE (One-Time Passwords in Everything) for PAM, allows remote attackers to determine the existence of certain user accounts, which displays a different response if the user account exists and is configured to use one-time passwords (OTP), a similar issue to CVE-2007-2243.
CVE-2007-4998	MEDIUM	N/A	N/A	linux-rittal 6.12.100+git	cp, when running with an option to preserve symlinks on multiple OSes, allows local, user-assisted attackers to overwrite arbitrary files via a symlink attack using crafted directories containing multiple source files that are copied to the same destination.
CVE-1999-0656	MEDIUM	N/A	N/A	linux-rittal 6.12.100+git	The ugidd RPC interface, by design, allows remote attackers to enumerate valid usernames by specifying arbitrary UIDs that ugidd maps to local user and group names.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-21807	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: block: fix queue freeze vs limits lock order in sysfs store methods queue_attr_store() always freezes a device queue before calling the attribute store operation. For attributes that control queue limits, the store operation will also lock the queue limits with a call to queue_limits_start_update(). However, some drivers (e.g. SCSI sd) may need to issue commands to a device to obtain limit values from the hardware with the queue limits locked. This creates a potential ABBA deadlock situation if a user attempts to modify a limit (thus freezing the device queue) while the device driver starts a revalidation of the device queue limits. Avoid such deadlock by not freezing the queue before calling the ->store_limit() method in struct queue_sysfs_entry and instead use the queue_limits_commit_update_frozen helper to freeze the queue after taking the limits lock. This also removes taking the sysfs lock for the store_limit method as it doesn't protect anything here, but creates even more nesting. Hopefully it will go away from the actual sysfs methods entirely soon. (commit log adapted from a similar patch from Damien Le Moal)

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-22111	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: Remove RTNL dance for SIOCBRADDIF and SIOCBRDELIF. SIOCBRDELIF is passed to dev_ioctl() first and later forwarded to br_ioctl_call(), which causes unnecessary RTNL dance and the splat below [0] under RTNL pressure. Let's say Thread A is trying to detach a device from a bridge and Thread B is trying to remove the bridge. In dev_ioctl(), Thread A bumps the bridge device's refcnt by netdev_hold() and releases RTNL because the following br_ioctl_call() also re-acquires RTNL. In the race window, Thread B could acquire RTNL and try to remove the bridge device. Then, rtnl_unlock() by Thread B will release RTNL and wait for netdev_put() by Thread A. Thread A, however, must hold RTNL after the unlock in dev_ifsioc(), which may take long under RTNL pressure, resulting in the splat by Thread B. Thread A (SIOCBRDELIF) Thread B (SIOCBRDELBR) ---------- ---------- -- sock_ioctl sock_ioctl `- sock_do_ioctl `- br_ioctl_call `- dev_ioctl `- br_ioctl_stub |- rtnl_lock	|- dev_ifsioc ' ' |- dev = __dev_get_by_name(...) |- netdev_hold(dev, ...) . / |- rtnl_unlock -------.		|- br_ioctl_call `---> |- rtnl_lock Race		`- br_ioctl_stub |- br_del_bridge Window			|- dev = __dev_get_by_name(...)			May take long	`- br_dev_delete(dev, ...)			under RTNL pressure	`- unregister_netdevice_queue(dev, ...)				`- rtnl_unlock \	|- rtnl_lock <-' `- netdev_run_todo	|- ... `- netdev_run_todo	`- rtnl_unlock |- __rtnl_unlock	|- netdev_wait_allrefs_any |- netdev_put(dev, ...) <---------------' Wait refcnt decrement and log splat below To avoid blocking SIOCBRDELBR unnecessarily, let's not call dev_ioctl() for SIOCBRADDIF and SIOCBRDELIF. In the dev_ioctl() path, we do the following: 1. Copy struct ifreq by get_user_ifreq in sock_do_ioctl() 2. Check CAP_NET_ADMIN in dev_ioctl() 3. Call dev_load() in dev_ioctl() 4. Fetch the master dev from ifr.ifr_name in dev_ifsioc() 3. can be done by request_module() in br_ioctl_call(), so we move 1., 2., and 4. to br_ioctl_stub(). Note that 2. is also checked later in add_del_if(), but it's better performed before RTNL. SIOCBRADDIF and SIOCBRDELIF have been processed in dev_ioctl() since the pre-git era, and there seems to be no specific reason to process them there. [0]: unregister_netdevice: waiting for wpan3 to become free. Usage count = 2 ref_tracker: wpan3@ffff8880662d8608 has 1/1 users at __netdev_tracker_alloc include/linux/netdevice.h:4282 [inline] netdev_hold include/linux/netdevice.h:4311 [inline] dev_ifsioc+0xc6a/0x1160 net/core/dev_ioctl.c:624 dev_ioctl+0x255/0x10c0 net/core/dev_ioctl.c:826 sock_do_ioctl+0x1ca/0x260 net/socket.c:1213 sock_ioctl+0x23a/0x6c0 net/socket.c:1318 vfs_ioctl fs/ioctl.c:51 [inline] __do_sys_ioctl fs/ioctl.c:906 [inline] __se_sys_ioctl fs/ioctl.c:892 [inline] __x64_sys_ioctl+0x1a4/0x210 fs/ioctl.c:892 do_syscall_x64 arch/x86/entry/common.c:52 [inline] do_syscall_64+0xcb/0x250 arch/x86/entry/common.c:83 entry_SYSCALL_64_after_hwframe+0x77/0x7f

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-38591	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: Reject narrower access to pointer ctx fields The following BPF program, simplified from a syzkaller repro, causes a kernel warning: r0 = *(u8 *)(r1 + 169); exit; With pointer field sk being at offset 168 in __sk_buff. This access is detected as a narrower read in bpf_skb_is_valid_access because it doesn't match offsetof(struct __sk_buff, sk). It is therefore allowed and later proceeds to bpf_convert_ctx_access. Note that for the "is_narrower_load" case in the convert_ctx_accesses(), the insn->off is aligned, so the cnt may not be 0 because it matches the offsetof(struct __sk_buff, sk) in the bpf_convert_ctx_access. However, the target_size stays 0 and the verifier errors with a kernel warning: verifier bug: error during ctx access conversion(1) This patch fixes that to return a proper "invalid bpf_context access off=X size=Y" error on the load instruction. The same issue affects multiple other fields in context structures that allow narrow access. Some other non-affected fields (for sk_msg, sk_lookup, and sockopt) were also changed to use bpf_ctx_range_ptr for consistency. Note this syzkaller crash was reported in the "Closes" link below, which used to be about a different bug, fixed in commit fce7bd8e385a ("bpf/verifier: Handle BPF_LOAD_ACQ instructions in insn_def_regno()"). Because syzbot somehow confused the two bugs, the new crash and repro didn't get reported to the mailing list.
CVE-2025-39764	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: ctnetlink: remove refcounting in expectation dumpers Same pattern as previous patch: do not keep the expectation object alive via refcount, only store a cookie value and then use that as the skip hint for dump resumption. AFAICS this has the same issue as the one resolved in the conntrack dumper, when we do if (!refcount_inc_not_zero(&exp->use)) to increment the refcount, there is a chance that exp == last, which causes a double-increment of the refcount and subsequent memory leak.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-39775	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: mm/mremap: fix WARN with uffd that has remap events disabled Registering userfaultd on a VMA that spans at least one PMD and then mremap()'ing that VMA can trigger a WARN when recovering from a failed page table move due to a page table allocation error. The code ends up doing the right thing (recurse, avoiding moving actual page tables), but triggering that WARN is unpleasant: WARNING: CPU: 2 PID: 6133 at mm/mremap.c:357 move_normal_pmd mm/mremap.c:357 [inline] WARNING: CPU: 2 PID: 6133 at mm/mremap.c:357 move_pgt_entry mm/mremap.c:595 [inline] WARNING: CPU: 2 PID: 6133 at mm/mremap.c:357 move_page_tables+0x3832/0x44a0 mm/mremap.c:852 Modules linked in: CPU: 2 UID: 0 PID: 6133 Comm: syz.0.19 Not tainted 6.17.0-rc1-syzkaller-00004-g53e760d89498 #0 PREEMPT(full) Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.16.3-debian-1.16.3-2~bpo12+1 04/01/2014 RIP: 0010:move_normal_pmd mm/mremap.c:357 [inline] RIP: 0010:move_pgt_entry mm/mremap.c:595 [inline] RIP: 0010:move_page_tables+0x3832/0x44a0 mm/mremap.c:852 Code: ... RSP: 0018:ffffc900037a76d8 EFLAGS: 00010293 RAX: 0000000000000000 RBX: 0000000032930007 RCX: ffffffff820c6645 RDX: ffff88802e56a440 RSI: ffffffff820c7201 RDI: 0000000000000007 RBP: ffff888037728fc0 R08: 0000000000000007 R09: 0000000000000000 R10: 0000000032930007 R11: 0000000000000000 R12: 0000000000000000 R13: ffffc900037a79a8 R14: 0000000000000001 R15: dffffc0000000000 FS: 000055556316a500(0000) GS:ffff8880d68bc000(0000) knlGS:0000000000000000 CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033 CR2: 0000001b30863fff CR3: 0000000050171000 CR4: 0000000000352ef0 Call Trace: copy_vma_and_data+0x468/0x790 mm/mremap.c:1215 move_vma+0x548/0x1780 mm/mremap.c:1282 mremap_to+0x1b7/0x450 mm/mremap.c:1406 do_mremap+0xfad/0x1f80 mm/mremap.c:1921 __do_sys_mremap+0x119/0x170 mm/mremap.c:1977 do_syscall_x64 arch/x86/entry/syscall_64.c:63 [inline] do_syscall_64+0xcd/0x4c0 arch/x86/entry/syscall_64.c:94 entry_SYSCALL_64_after_hwframe+0x77/0x7f RIP: 0033:0x7f00d0b8ebe9 Code: ... RSP: 002b:00007ffe5ea5ee98 EFLAGS: 00000246 ORIG_RAX: 0000000000000019 RAX: ffffffffffffffda RBX: 00007f00d0db5fa0 RCX: 00007f00d0b8ebe9 RDX: 0000000000400000 RSI: 0000000000c00000 RDI: 0000200000000000 RBP: 00007ffe5ea5eef0 R08: 0000200000c00000 R09: 0000000000000000 R10: 0000000000000003 R11: 0000000000000246 R12: 0000000000000002 R13: 00007f00d0db5fa0 R14: 00007f00d0db5fa0 R15: 0000000000000005 The underlying issue is that we recurse during the original page table move, but not during the recovery move. Fix it by checking for both VMAs and performing the check before the pmd_none() sanity check. Add a new helper where we perform+document that check for the PMD and PUD level. Thanks to Harry for bisecting.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-39822	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: io_uring/kbuf: fix signedness in this_len calculation When importing and using buffers, buf->len is considered unsigned. However, buf->len is converted to signed int when committing. This can lead to unexpected behavior if the buffer is large enough to be interpreted as a negative value. Make min_t calculation unsigned.
CVE-2025-68351	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: exfat: fix refcount leak in exfat_find Fix refcount leaks in `exfat_find` related to `exfat_get_dentry_set`. Function `exfat_get_dentry_set` would increase the reference counter of `es->bh` on success. Therefore, `exfat_put_dentry_set` must be called after `exfat_get_dentry_set` to ensure refcount consistency. This patch relocate two checks to avoid possible leaks.
CVE-2025-71117	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: block: Remove queue freezing from several sysfs store callbacks Freezing the request queue from inside sysfs store callbacks may cause a deadlock in combination with the dm-multipath driver and the queue_if_no_path option. Additionally, freezing the request queue slows down system boot on systems where sysfs attributes are set synchronously. Fix this by removing the blk_mq_freeze_queue() / blk_mq_unfreeze_queue() calls from the store callbacks that do not strictly need these callbacks. Add the __data_racy annotation to request_queue.rq_timeout to suppress KCSAN data race reports about the rq_timeout reads. This patch may cause a small delay in applying the new settings. For all the attributes affected by this patch, I/O will complete correctly whether the old or the new value of the attribute is used. This patch affects the following sysfs attributes: * io_poll_delay * io_timeout * nomerges * read_ahead_kb * rq_affinity Here is an example of a deadlock triggered by running test srp/002 if this patch is not applied: task:multipathd Call Trace: __schedule+0x8c1/0x1bf0 schedule+0xdd/0x270 schedule_preempt_disabled+0x1c/0x30 __mutex_lock+0xb89/0x1650 mutex_lock_nested+0x1f/0x30 dm_table_set_restrictions+0x823/0xdf0 __bind+0x166/0x590 dm_swap_table+0x2a7/0x490 do_resume+0x1b1/0x610 dev_suspend+0x55/0x1a0 ctl_ioctl+0x3a5/0x7e0 dm_ctl_ioctl+0x12/0x20 __x64_sys_ioctl+0x127/0x1a0 x64_sys_call+0xe2b/0x17d0 do_syscall_64+0x96/0x3a0 entry_SYSCALL_64_after_hwframe+0x4b/0x53 task:(udev-worker) Call Trace: __schedule+0x8c1/0x1bf0 schedule+0xdd/0x270 blk_mq_freeze_queue_wait+0xf2/0x140 blk_mq_freeze_queue_nomemsave+0x23/0x30 queue_ra_store+0x14e/0x290 queue_attr_store+0x23e/0x2c0 sysfs_kf_write+0xde/0x140 kernfs_fop_write_iter+0x3b2/0x630 vfs_write+0x4fd/0x1390 ksys_write+0xfd/0x230 __x64_sys_write+0x76/0xc0 x64_sys_call+0x276/0x17d0 do_syscall_64+0x96/0x3a0 entry_SYSCALL_64_after_hwframe+0x4b/0x53

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2025-71141	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/tilcdc: Fix removal actions in case of failed probe The drm_kms_helper_poll_fini() and drm_atomic_helper_shutdown() helpers should only be called when the device has been successfully registered. Currently, these functions are called unconditionally in tilcdc_fini(), which causes warnings during probe deferral scenarios. [7.972317] WARNING: CPU: 0 PID: 23 at drivers/gpu/drm/drm_atomic_state_helper.c:175 drm_atomic_helper_crtc_duplicate_state+0x60/0x68 ... [8.005820] drm_atomic_helper_crtc_duplicate_state from drm_atomic_get_crtc_state+0x68/0x108 [8.005858] drm_atomic_get_crtc_state from drm_atomic_helper_disable_all+0x90/0x1c8 [8.005885] drm_atomic_helper_disable_all from drm_atomic_helper_shutdown+0x90/0x144 [8.005911] drm_atomic_helper_shutdown from tilcdc_fini+0x68/0xf8 [tilcdc] [8.005957] tilcdc_fini [tilcdc] from tilcdc_pdev_probe+0xb0/0x6d4 [tilcdc] Fix this by rewriting the failed probe cleanup path using the standard goto error handling pattern, which ensures that cleanup functions are only called on successfully initialized resources. Additionally, remove the now-unnecessary is_registered flag.
CVE-2026-22986	MEDIUM	4.7	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: gpiolib: fix race condition for gdev->srcu If two drivers were calling gpiochip_add_data_with_key(), one may be traversing the srcu-protected list in gpio_name_to_desc(), meanwhile other has just added its gdev in gpiodev_add_to_list_unlocked(). This creates a non-mutexed and non-protected timeframe, when one instance is dereferencing and using &gdev->srcu, before the other has initialized it, resulting in crash: [4.935481] Unable to handle kernel paging request at virtual address ffff800272bcc000 [4.943396] Mem abort info: [4.943400] ESR = 0x0000000096000005 [4.943403] EC = 0x25: DABT (current EL), IL = 32 bits [4.943407] SET = 0, FnV = 0 [4.943410] EA = 0, S1PTW = 0 [4.943413] FSC = 0x05: level 1 translation fault [4.943416] Data abort info: [4.943418] ISV = 0, ISS = 0x00000005, ISS2 = 0x00000000 [4.946220] CM = 0, WnR = 0, TnD = 0, TagAccess = 0 [4.955261] GCS = 0, Overlay = 0, DirtyBit = 0, Xs = 0 [4.955268] swapper pgtable: 4k pages, 48-bit VAs, pgdp=0000000038e6c000 [4.961449] [ffff800272bcc000] pgd=0000000000000000 [4.969203] , p4d=1000000039739003 [4.979730] , pud=0000000000000000 [4.980210] phandle (CPU): 0x0000005e, phandle (BE): 0x5e000000 for node "reset" [4.991736] Internal error: Oops: 0000000096000005 [#1] PREEMPT SMP ... [5.121359] pc : __srcu_read_lock+0x44/0x98 [5.131091] lr : gpio_name_to_desc+0x60/0x1a0 [5.153671] sp : ffff8000833bb430 [5.298440] [5.298443] Call trace: [5.298445] __srcu_read_lock+0x44/0x98 [5.309484] gpio_name_to_desc+0x60/0x1a0 [5.320692] gpiochip_add_data_with_key+0x488/0xf00 5.946419] ---[end trace 0000000000000000]--- Move initialization code for gdev fields before it is added to gpio_devices, with adjacent initialization code. Adjust goto statements to reflect modified order of operations [Bartosz: fixed a build issue, removed stray newline]

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-23007	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: block: zero non-PI portion of auto integrity buffer The auto-generated integrity buffer for writes needs to be fully initialized before being passed to the underlying block device, otherwise the uninitialized memory can be read back by userspace or anyone with physical access to the storage device. If protection information is generated, that portion of the integrity buffer is already initialized. The integrity data is also zeroed if PI generation is disabled via sysfs or the PI tuple size is 0. However, this misses the case where PI is generated and the PI tuple size is nonzero, but the metadata size is larger than the PI tuple. In this case, the remainder ("opaque") of the metadata is left uninitialized. Generalize the BLK_INTEGRITY_CSUM_NONE check to cover any case when the metadata is larger than just the PI tuple.
CVE-2026-23138	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: tracing: Add recursion protection in kernel stack trace recording A bug was reported about an infinite recursion caused by tracing the rcu events with the kernel stack trace trigger enabled. The stack trace code called back into RCU which then called the stack trace again. Expand the ftrace recursion protection to add a set of bits to protect events from recursion. Each bit represents the context that the event is in (normal, softirq, interrupt and NMI). Have the stack trace code use the interrupt context to protect against recursion. Note, the bug showed an issue in both the RCU code as well as the tracing stacktrace code. This only handles the tracing stack trace side of the bug. The RCU fix will be handled separately.
CVE-2026-23247	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: tcp: secure_seq: add back ports to TS offset This reverts 28ee1b746f49 ("secure_seq: downgrade to per-host timestamp offsets") tcp_tw_recycle went away in 2017. Zhouyan Deng reported off-path TCP source port leakage via SYN cookie side-channel that can be fixed in multiple ways. One of them is to bring back TCP ports in TS offset randomization. As a bonus, we perform a single siphash() computation to provide both an ISN and a TS offset.
CVE-2026-23259	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: io_uring/rw: free potentially allocated iovec on cache put failure If a read/write request goes through io_req_rw_cleanup() and has an allocated iovec attached and fails to put to the rw_cache, then it may end up with an unaccounted iovec pointer. Have io_rw_recycle() return whether it recycled the request or not, and use that to gauge whether to free a potential iovec or not.
CVE-2026-29111	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	systemd 255.21	systemd, a system and service manager, (as PID 1) hits an assert and freezes execution when an unprivileged IPC API call is made with spurious data. On version v249 and older the effect is not an assert, but stack overwriting, with the attacker controlled content. From version v250 and newer this is not possible as the safety check causes an assert instead. This IPC call was added in v239, so versions older than that are not affected. Versions 260-rc1, 259.2, 258.5, and 257.11 contain patches. No known workarounds are available.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-23346	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: arm64: io: Extract user memory type in ioremap_prot() The only caller of ioremap_prot() outside of the generic ioremap() implementation is generic_access_phys(), which passes a 'pgprot_t' value determined from the user mapping of the target 'pfn' being accessed by the kernel. On arm64, the 'pgprot_t' contains all of the non-address bits from the pte, including the permission controls, and so we end up returning a new user mapping from ioremap_prot() which faults when accessed from the kernel on systems with PAN: | Unable to handle kernel read from unreadable memory at virtual address ffff80008ea89000 | ... | Call trace: | __memcpy_fromio+0x80/0xf8 | generic_access_phys+0x20c/0x2b8 | __access_remote_vm+0x46c/0x5b8 | access_remote_vm+0x18/0x30 | environ_read+0x238/0x3e8 | vfs_read+0xe4/0x2b0 | ksys_read+0xcc/0x178 | __arm64_sys_read+0x4c/0x68 Extract only the memory type from the user 'pgprot_t' in ioremap_prot() and assert that we're being passed a user mapping, to protect us against any changes in future that may require additional handling. To avoid falsely flagging users of ioremap(), provide our own ioremap() macro which simply wraps __ioremap_prot().

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-23385	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: nf_tables: clone set on flush only Syzbot with fault injection triggered a failing memory allocation with GFP_KERNEL which results in a WARN splat: iter.err WARNING: net/netfilter/nf_tables_api.c:845 at nft_map_deactivate+0x34e/0x3c0 net/netfilter/nf_tables_api.c:845, CPU#0: syz.0.17/5992 Modules linked in: CPU: 0 UID: 0 PID: 5992 Comm: syz.0.17 Not tainted syzkaller #0 PREEMPT(full) Hardware name: Google Google Compute Engine/Google Compute Engine, BIOS Google 02/12/2026 RIP: 0010:nft_map_deactivate+0x34e/0x3c0 net/netfilter/nf_tables_api.c:845 Code: 8b 05 86 5a 4e 09 48 3b 84 24 a0 00 00 00 75 62 48 8d 65 d8 5b 41 5c 41 5d 41 5e 41 5f 5d c3 cc cc cc cc cc e8 63 6d fa f7 90 <0f> 0b 90 43 +80 7c 35 00 00 0f 85 23 fe ff ff e9 26 fe ff ff 89 d9 RSP: 0018:ffffc900045af780 EFLAGS: 00010293 RAX: ffffffff89ca45bd RBX: 00000000ffffffff4 RCX: ffff888028111e40 RDX: 0000000000000000 RSI: 00000000ffffffff4 RDI: 0000000000000000 RBP: ffffc900045af870 R08: 0000000000400dc0 R09: 00000000ffffffff R10: dffffc0000000000 R11: fffffbfff1d141db R12: ffffc900045af7e0 R13: 1ffff920008b5f24 R14: dffffc0000000000 R15: ffffc900045af920 FS: 000055557a6a5500(0000) GS:ffff888125496000(0000) knlGS:0000000000000000 CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033 CR2: 00007fb5ea271fc0 CR3: 000000003269e000 CR4: 00000000003526f0 Call Trace: __nft_release_table+0xceb/0x11f0 net/netfilter/nf_tables_api.c:12115 nft_rcv_nl_event+0xc25/0xdb0 net/netfilter/nf_tables_api.c:12187 notifier_call_chain+0x19d/0x3a0 kernel/notifier.c:85 blocking_notifier_call_chain+0x6a/0x90 kernel/notifier.c:380 netlink_release+0x123b/0x1ad0 net/netlink/af_netlink.c:761 __sock_release net/socket.c:662 [inline] sock_close+0xc3/0x240 net/socket.c:1455 Restrict set clone to the flush set command in the preparation phase. Add NFT_ITER_UPDATE_CLONE and use it for this purpose, update the rbtree and pipapo backends to only clone the set when this iteration type is used. As for the existing NFT_ITER_UPDATE type, update the pipapo backend to use the existing set clone if available, otherwise use the existing set representation. After this update, there is no need to clone a set that is being deleted, this includes bound anonymous set. An alternative approach to NFT_ITER_UPDATE_CLONE is to add a .clone interface and call it from the flush set path.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-23394	MEDIUM	4.7	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: af_unix: Give up GC if MSG_PEEK intervened. Igor Ushakov reported that GC purged the receive queue of an alive socket due to a race with MSG_PEEK with a nice repro. This is the exact same issue previously fixed by commit cbcf01128d0a ("af_unix: fix garbage collect vs MSG_PEEK"). After GC was replaced with the current algorithm, the cited commit removed the locking dance in unix_peek_fds() and reintroduced the same issue. The problem is that MSG_PEEK bumps a file refcount without interacting with GC. Consider an SCC containing sk-A and sk-B, where sk-A is close()d but can be recv()ed via sk-B. The bad thing happens if sk-A is recv()ed with MSG_PEEK from sk-B and sk-B is close()d while GC is checking unix_vertex_dead() for sk-A and sk-B. GC thread User thread --------- ---------- unix_vertex_dead(sk-A) -> true <------. \ `------ recv(sk-B, MSG_PEEK) invalidate !! -> sk-A's file refcount : 1 -> 2 close(sk-B) -> sk-B's file refcount : 2 -> 1 unix_vertex_dead(sk-B) -> true Initially, sk-A's file refcount is 1 by the inflight fd in sk-B recvq. GC thinks sk-A is dead because the file refcount is the same as the number of its inflight fds. However, sk-A's file refcount is bumped silently by MSG_PEEK, which invalidates the previous evaluation. At this moment, sk-B's file refcount is 2; one by the open fd, and one by the inflight fd in sk-A. The subsequent close() releases one refcount by the former. Finally, GC incorrectly concludes that both sk-A and sk-B are dead. One option is to restore the locking dance in unix_peek_fds(), but we can resolve this more elegantly thanks to the new algorithm. The point is that the issue does not occur without the subsequent close() and we actually do not need to synchronise MSG_PEEK with the dead SCC detection. When the issue occurs, close() and GC touch the same file refcount. If GC sees the refcount being decremented by close(), it can just give up garbage-collecting the SCC. Therefore, we only need to signal the race during MSG_PEEK with a proper memory barrier to make it visible to the GC. Let's use seqcount_t to notify GC when MSG_PEEK occurs and let it defer the SCC to the next run. This way no locking is needed on the MSG_PEEK side, and we can avoid imposing a penalty on every MSG_PEEK unnecessarily. Note that we can retry within unix_scc_dead() if MSG_PEEK is detected, but we do not do so to avoid hung task splat from abusive MSG_PEEK calls.
CVE-2026-35414	MEDIUM	4.2	CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:L/A:N	openssh 9.6p1	OpenSSH before 10.3 mishandles the authorized_keys principals option in uncommon scenarios involving a principals list in conjunction with a Certificate Authority that makes certain use of comma characters.
CVE-2026-23472	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: serial: core: fix infinite loop in handle_tx() for PORT_UNKNOWN uart_write_room() and uart_write() behave inconsistently when xmit_buf is NULL (which happens for PORT_UNKNOWN ports that were never properly initialized): - uart_write_room() returns kfifo_avail() which can be > 0 - uart_write() checks xmit_buf and returns 0 if NULL This inconsistency causes an infinite loop in drivers that rely on tty_write_room() to determine if they can write: while (tty_write_room(tty) > 0) { written = tty->ops->write(...); // written is always 0, loop never exits } For example, caif_serial's handle_tx() enters an infinite loop when used with PORT_UNKNOWN serial ports, causing system hangs. Fix by making uart_write_room() also check xmit_buf and return 0 if it's NULL, consistent with uart_write(). Reproducer: https://gist.github.com/mrpre/d9a694cc0e19828ee3bc3b37983fde13

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-31456	MEDIUM	4.7	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: mm/pagewalk: fix race between concurrent split and refault The splitting of a PUD entry in walk_pud_range() can race with a concurrent thread refaulting the PUD leaf entry causing it to try walking a PMD range that has disappeared. An example and reproduction of this is to try reading numa_maps of a process while VFIO-PCI is setting up DMA (specifically the vfio_pin_pages_remote call) on a large BAR for that process. This will trigger a kernel BUG: vfio-pci 0000:03:00.0: enabling device (0000 -> 0002) BUG: unable to handle page fault for address: ffffa23980000000 PGD 0 P4D 0 Oops: Oops: 0000 [#1] SMP NOPTI ... RIP: 0010:walk_pgd_range+0x3b5/0x7a0 Code: 8d 43 ff 48 89 44 24 28 4d 89 ce 4d 8d a7 00 00 20 00 48 8b 4c 24 28 49 81 e4 00 00 e0 ff 49 8d 44 24 ff 48 39 c8 4c 0f 43 e3 <49> f7 06 9f ff ff ff 75 3b 48 8b 44 24 20 48 8b 40 28 48 85 c0 74 RSP: 0018:ffffac23e1ecf808 EFLAGS: 00010287 RAX: 00007f44c01fffff RBX: 00007f4500000000 RCX: 00007f44ffffffff RDX: 0000000000000000 RSI: 000fffffffff000 RDI: ffffffff93378fe0 RBP: ffffac23e1ecf918 R08: 0000000000000004 R09: ffffa23980000000 R10: 0000000000000020 R11: 0000000000000004 R12: 00007f44c0200000 R13: 00007f44c0000000 R14: ffffa23980000000 R15: 00007f44c0000000 FS: 00007fe884739580(0000) GS:ffff9b7d7a9c0000(0000) knlGS:0000000000000000 CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033 CR2: ffffa23980000000 CR3: 000000c0650e2005 CR4: 0000000000770ef0 PKRU: 55555554 Call Trace: __walk_page_range+0x195/0x1b0 walk_page_vma+0x62/0xc0 show_numa_map+0x12b/0x3b0 seq_read_iter+0x297/0x440 seq_read+0x11d/0x140 vfs_read+0xc2/0x340 ksys_read+0x5f/0xe0 do_syscall_64+0x68/0x130 ? get_page_from_freelist+0x5c2/0x17e0 ? mas_store_prealloc+0x17e/0x360 ? vma_set_page_prot+0x4c/0xa0 ? __alloc_pages_noprof+0x14e/0x2d0 ? __mod_memcg_lruvec_state+0x8d/0x140 ? __lruvec_stat_mod_folio+0x76/0xb0 ? __folio_mod_stat+0x26/0x80 ? do_anonymous_page+0x705/0x900 ? __handle_mm_fault+0xa8d/0x1000 ? __count_memcg_events+0x53/0xf0 ? handle_mm_fault+0xa5/0x360 ? do_user_addr_fault+0x342/0x640 ? arch_exit_to_user_mode_prepare.constprop.0+0x16/0xa0 ? irqentry_exit_to_user_mode+0x24/0x100 entry_SYSCALL_64_after_hwframe+0x76/0x7e RIP: 0033:0x7fe88464f47e Code: c0 e9 b6 fe ff ff 50 48 8d 3d be 07 0b 00 e8 69 01 02 00 66 0f 1f 84 00 00 00 00 00 64 8b 04 25 18 00 00 00 85 c0 75 14 0f 05 <48> 3d 00 f0 ff ff 77 5a c3 66 0f 1f 84 00 00 00 00 00 48 83 ec 28 RSP: 002b:00007ffe6cd9a9b8 EFLAGS: 00000246 ORIG_RAX: 0000000000000000 RAX: ffffffffffffffda RBX: 0000000000020000 RCX: 00007fe88464f47e RDX: 0000000000020000 RSI: 00007fe884543000 RDI: 0000000000000003 RBP: 00007fe884543000 R08: 00007fe884542010 R09: 0000000000000000 R10: ffffffffffffffbc5 R11: 0000000000000246 R12: 0000000000000000 R13: 0000000000000003 R14: 0000000000020000 R15: 0000000000020000 Fix this by validating the PUD entry in walk_pmd_range() using a stable snapshot (pudp_get()). If the PUD is not present or is a leaf, retry the walk via ACTION_AGAIN instead of descending further. This mirrors the retry logic in walk_pte_range(), which lets walk_pmd_range() retry if the PTE is not being got by pte_offset_map_lock().

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-6238	MEDIUM	6.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:L	glibc 2.39+git	The deprecated functions ns_printrrf, ns_printrr and fp_nquery in the GNU C Library version 2.0.1 to version 2.43 fail to validate the RDATA content against the RDATA length in a DNS response when processing A6, CERT, LOC, TKEY or TSIG records, which may allow an attacker to craft a DNS response, causing a target application to crash or read uninitialized memory. These functions are for application debugging only and hence not in the path of code executed by the DNS resolver. Further, they have been deprecated since version 2.34 and should not be used by any new applications. Applications should consider porting away from these interfaces since they may be removed in future versions.
CVE-2026-31692	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: rtnetlink: add missing netlink_ns_capable() check for peer netns rtnl_newlink() lacks a CAP_NET_ADMIN capability check on the peer network namespace when creating paired devices (veth, vxcan, netkit). This allows an unprivileged user with a user namespace to create interfaces in arbitrary network namespaces, including init_net. Add a netlink_ns_capable() check for CAP_NET_ADMIN in the peer namespace before allowing device creation to proceed.
CVE-2026-31732	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: gpio: Fix resource leaks on errors in gpiochip_add_data_with_key() Since commit aab5c6f20023 ("gpio: set device type for GPIO chips"), `gdev->dev.release` is unset. As a result, the reference count to `gdev->dev` isn't dropped on the error handling paths. Drop the reference on errors. Also reorder the instructions to make the error handling simpler. Now gpiochip_add_data_with_key() roughly looks like: >>> Some memory allocation. Go to ERR ZONE 1 on errors. >>> device_initialize(). gpiodev_release() takes over the responsibility for freeing the resources of `gdev->dev`. The subsequent error handling paths shouldn't go through ERR ZONE 1 again which leads to double free. >>> Some initialization mainly on `gdev`. >>> The rest of initialization. Go to ERR ZONE 2 on errors. >>> Chip registration success and exit. >>> ERR ZONE 2. gpio_device_put() and exit. >>> ERR ZONE 1.
CVE-2026-43109	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: x86: shadow stacks: proper error handling for mmap lock 김영민 reports that shstk_pop_sigframe() doesn't check for errors from mmap_read_lock_killable(), which is a silly oversight, and also shows that we haven't marked those functions with "__must_check", which would have immediately caught it. So let's fix both issues.
CVE-2026-43127	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ntfs3: fix circular locking dependency in run_unpack_ex Syzbot reported a circular locking dependency between wnd->rw_lock (sbi->used.bitmap) and ni->file.run_lock. The deadlock scenario: 1. ntfs_extend_mft() takes ni->file.run_lock then wnd->rw_lock. 2. run_unpack_ex() takes wnd->rw_lock then tries to acquire ni->file.run_lock inside ntfs_refresh_zone(). This creates an AB-BA deadlock. Fix this by using down_read_trylock() instead of down_read() when acquiring run_lock in run_unpack_ex(). If the lock is contended, skip ntfs_refresh_zone() - the MFT zone will be refreshed on the next MFT operation. This breaks the circular dependency since we never block waiting for run_lock while holding wnd->rw_lock.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-43216	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: Drop the lock in skb_may_tx_timestamp() skb_may_tx_timestamp() may acquire sock::sk_callback_lock. The lock must not be taken in IRQ context, only softirq is okay. A few drivers receive the timestamp via a dedicated interrupt and complete the TX timestamp from that handler. This will lead to a deadlock if the lock is already write-locked on the same CPU. Taking the lock can be avoided. The socket (pointed by the skb) will remain valid until the skb is released. The ->sk_socket and ->file member will be set to NULL once the user closes the socket which may happen before the timestamp arrives. If we happen to observe the pointer while the socket is closing but before the pointer is set to NULL then we may use it because both pointer (and the file's cred member) are RCU freed. Drop the lock. Use READ_ONCE() to obtain the individual pointer. Add a matching WRITE_ONCE() where the pointer are cleared.
CVE-2026-4873	MEDIUM	5.9	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N	curl 8.7.1	A vulnerability exists where a connection requiring TLS incorrectly reuses an existing unencrypted connection from the same connection pool. If an initial transfer is made in clear-text (via IMAP, SMTP, or POP3), a subsequent request to that same host bypasses the TLS requirement and instead transmit data unencrypted.
CVE-2026-5545	MEDIUM	6.5	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:H/A:N	curl 8.7.1	libcurl might in some circumstances reuse the wrong connection when asked to do an authenticated HTTP(S) request after a Negotiate-authenticated one, when both use the same host. libcurl features a pool of recent connections so that subsequent requests can reuse an existing connection to avoid overhead. When reusing a connection a range of criteria must be met. Due to a logical error in the code, a request that was issued by an application could wrongfully reuse an existing connection to the same server that was authenticated using different credentials. An application that first uses Negotiate authentication to a server with `user1:password1` and then does another operation to the same server asking for any authentication method but for `user2:password2` (while the previous connection is still alive) - the second request gets confused and wrongly reuses the same connection and sends the new request over that connection thinking it uses a mix of user1's and user2's credentials when it is in fact still using the connection authenticated for user1...
CVE-2026-6253	MEDIUM	5.9	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H	curl 8.7.1	curl might erroneously pass on credentials for a first proxy to a second proxy. This can happen when the following conditions are true: 1. curl is setup to use specific different proxies for different URL schemes 2. the first proxy needs credentials 3. the second proxy uses no credentials 4. while using the first proxy (using say `http://`), curl is asked to follow a redirect to a URL using another scheme (say `https://`) accessed using a second, different, proxy
CVE-2026-43485	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: nouveau/gsp: drop WARN_ON in ACPI probes These WARN_ONs seem to trigger a lot, and we don't seem to have a plan to fix them, so just drop them, as they are most likely harmless.
CVE-2026-45934	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: btrfs: fix EEXIST abort due to non-consecutive gaps in chunk allocation I have been observing a number of systems aborting at insert_dev_extents() in btrfs_create_pending_block_groups(). The following is a sample stack trace of such an abort coming from forced chunk allocation (typically behind

CVE	Severity	CVSS	CVSS Vector	Component	Description
					CONFIG_BTRFS_EXPERIMENTAL) but this can theoretically happen to any DUP chunk allocation. [81.801] ----------[cut here]---------- [81.801] BTRFS: Transaction aborted (error -17) [81.801] WARNING: fs/btrfs/block-group.c:2876 at btrfs_create_pending_block_groups+0x721/0x770 [btrfs], CPU#1: bash/319 [81.802] Modules linked in: virtio_net btrfs xor zstd_compress raid6_pq null_blk [81.803] CPU: 1 UID: 0 PID: 319 Comm: bash Kdump: loaded Not tainted 6.19.0-rc6+ #319 NONE [81.803] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Arch Linux 1.17.0-2-2 04/01/2014 [81.804] RIP: 0010:btrfs_create_pending_block_groups+0x723/0x770 [btrfs] [81.806] RSP: 0018:ffffa36241a6bce8 EFLAGS: 00010282 [81.806] RAX: 000000000000000d RBX: ffff8e699921e400 RCX: 0000000000000000 [81.807] RDX: 0000000002040001 RSI: 00000000ffffffef RDI: ffffffffc0608bf0 [81.807] RBP: 00000000ffffffef R08: ffff8e69830f6000 R09: 0000000000000007 [81.808] R10: ffff8e699921e5e8 R11: 0000000000000000 R12: ffff8e6999228000 [81.808] R13: ffff8e6984d82000 R14: ffff8e69966a69c0 R15: ffff8e69aa47b000 [81.809] FS: 00007fec6bdd9740(0000) GS:ffff8e6b1b379000(0000) knlGS:0000000000000000 [81.809] CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033 [81.810] CR2: 00005604833670f0 CR3: 0000000116679000 CR4: 00000000000006f0 [81.810] Call Trace: [81.810] [81.810] __btrfs_end_transaction+0x3e/0x2b0 [btrfs] [81.811] btrfs_force_chunk_alloc_store+0xcd/0x140 [btrfs] [81.811] kernfs_fop_write_iter+0x15f/0x240 [81.812] vfs_write+0x264/0x500 [81.812] ksys_write+0x6c/0xe0 [81.812] do_syscall_64+0x66/0x770 [81.812] entry_SYSCALL_64_after_hwframe+0x76/0x7e [81.813] RIP: 0033:0x7fec6be66197 [81.814] RSP: 002b:00007fffb159dd30 EFLAGS: 00000202 ORIG_RAX: 0000000000000001 [81.815] RAX: ffffffffffffffda RBX: 00007fec6bdd9740 RCX: 00007fec6be66197 [81.815] RDX: 0000000000000002 RSI: 00005604833374f80 RDI: 0000000000000001 [81.816] RBP: 00005604833374f80 R08: 0000000000000000 R09: 0000000000000000 [81.816] R10: 0000000000000000 R11: 0000000000000202 R12: 0000000000000002 [81.817] R13: 00007fec6bfb85c0 R14: 00007fec6bfb5ee0 R15: 00005604833729c0 [81.817] [81.817] irq event stamp: 20039 [81.818] hardirqs last enabled at (20047): [] __up_console_sem+0x52/0x60 [81.818] hardirqs last disabled at (20056): [] __up_console_sem+0x37/0x60 [81.819] softirqs last enabled at (19470): [] __irq_exit_rcu+0x96/0xc0 [81.819] softirqs last disabled at (19463): [] __irq_exit_rcu+0x96/0xc0 [81.820] ---[end trace 0000000000000000]--- [81.820] BTRFS: error (device dm-7 state A) in btrfs_create_pending_block_groups:2876: errno=-17 Object already exists Inspecting these aborts with drgn, I observed a pattern of overlapping chunk_maps. Note how stripe 1 of the first chunk overlaps in physical address with stripe 0 of the second chunk. Physical Start Physical End Length Logical Type Stripe ---------- ---------- ---------- ---------- ---------- ---------- 0x0000000102500000 0x0000000142500000 1.0G 0x0000000641d00000 META|DUP 0/2 0x0000000142500000 0x0000000182500000 1.0G 0x0000000641d00000 META|DUP 1/2 0x0000000142500000 0x0000000182500000 1.0G 0x0000000601d00000 META|DUP 0/2 0x0000000182500000 0x00000001c2500000 1.0G 0x0000000601d00000 META|DUP 1/2 Now how could this possibly happen? All chunk allocation is ---truncated---

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-45940	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: stmmac: fix oops when split header is enabled For GMAC4, when split header is enabled, in some rare cases, the hardware does not fill buf2 of the first descriptor with payload. Thus we cannot assume buf2 is always fully filled if it is not the last descriptor. Otherwise, the length of buf2 of the second descriptor will be calculated wrong and cause an oops: Unable to handle kernel paging request at virtual address ffff00019246bfc0 ... x2 : 0000000000000040 x1 : ffff00019246bfc0 x0 : ffff00009246c000 Call trace: dcache_inval_poc+0x28/0x58 (P) dma_direct_sync_single_for_cpu+0x38/0x6c __dma_sync_single_for_cpu+0x34/0x6c stmmac_napi_poll_rx+0x8f0/0xb60 __napi_poll.constprop.0+0x30/0x144 net_rx_action+0x160/0x274 handle_softirqs+0x1b8/0x1fc ... To fix this, the PL bit-field in RDES3 register is used for all descriptors, whether it is the last descriptor or not.
CVE-2026-46014	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: KVM: SVM: Add missing save/restore handling of LBR MSRs MSR_IA32_DEBUGCTLMSR and LBR MSRs are currently not enumerated by KVM_GET_MSR_INDEX_LIST, and LBR MSRs cannot be set with KVM_SET_MSRS. So save/restore is completely broken. Fix it by adding the MSRs to msrs_to_save_base, and allowing writes to LBR MSRs from userspace only (as they are read-only MSRs) if LBR virtualization is enabled. Additionally, to correctly restore L1's LBRs while L2 is running, make sure the LBRs are copied from the captured VMCB01 save area in svm_copy_vmrun_state(). Note, for VMX, this also fixes a flaw where MSR_IA32_DEBUGCTLMSR isn't reported as an MSR to save/restore. Note #2, over-reporting MSR_IA32_LASTxxx on Intel is ok, as KVM already handles unsupported reads and writes thanks to commit b5e2fec0ebc3 ("KVM: Ignore DEBUGCTL MSRs with no effect") (kvm_do_msr_access() will morph the unsupported userspace write into a nop). [sean: guard with lbrv checks, massage changelog]
CVE-2026-46044	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ipmi:ssif: Clean up kthread on errors If an error occurs after the ssif kthread is created, but before the main IPMI code starts the ssif interface, the ssif kthread will not be stopped. So make sure the kthread is stopped on an error condition if it is running.
CVE-2026-46059	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: KVM: nSVM: Always use NextRIP as vmcb02's NextRIP after first L2 VMRUN For guests with NRIPS disabled, L1 does not provide NextRIP when running an L2 with an injected soft interrupt, instead it advances the current RIP before running it. KVM uses the current RIP as the NextRIP in vmcb02 to emulate a CPU without NRIPS. However, after L2 runs the first time, NextRIP will be updated by the CPU and/or KVM, and the current RIP is no longer the correct value to use in vmcb02. Hence, after save/restore, use the current RIP if and only if a nested run is pending, otherwise use NextRIP. Give soft_int_next_rip the same treatment, as it's the same logic, just for a narrower use case. [sean: give soft_int_next_rip the same treatment]

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-46066	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ceph: fix num_ops off-by-one when crypto allocation fails move_dirty_folio_in_page_array() may fail if the file is encrypted, the dirty folio is not the first in the batch, and it fails to allocate a bounce buffer to hold the ciphertext. When that happens, ceph_process_folio_batch() simply redirties the folio and flushes the current batch -- it can retry that folio in a future batch. However, if this failed folio is not contiguous with the last folio that did make it into the batch, then ceph_process_folio_batch() has already incremented `ceph_wbc->num_ops`; because it doesn't follow through and add the discontiguous folio to the array, ceph_submit_write() -- which expects that `ceph_wbc->num_ops` accurately reflects the number of contiguous ranges (and therefore the required number of "write extent" ops) in the writeback -- will panic the kernel: BUG_ON(ceph_wbc->op_idx + 1 != req->r_num_ops); This issue can be reproduced on affected kernels by writing to fscrypt-enabled CephFS file(s) with a 4KiB-written/4KiB-skipped/repeat pattern (total filesize should not matter) and gradually increasing the system's memory pressure until a bounce buffer allocation fails. Fix this crash by decrementing `ceph_wbc->num_ops` back to the correct value when move_dirty_folio_in_page_array() fails, but the folio already started counting a new (i.e. still-empty) extent. The defect corrected by this patch has existed since 2022 (see first `Fixes:`), but another bug blocked multi-folio encrypted writeback until recently (see second `Fixes:`). The second commit made it into 6.18.16, 6.19.6, and 7.0-rc1, unmasking the panic in those versions. This patch therefore fixes a regression (panic) introduced by cac190c7674f.
CVE-2026-46071	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: KVM: nSVM: Avoid clearing VMCB_LBR in vmcb12 svm_copy_lbrs() always marks VMCB_LBR dirty in the destination VMCB. However, nested_svm_vmexit() uses it to copy LBRs to vmcb12, and clearing clean bits in vmcb12 is not architecturally defined. Move vmcb_mark_dirty() to callers and drop it for vmcb12. This also facilitates incoming refactoring that does not pass the entire VMCB to svm_copy_lbrs().
CVE-2026-46096	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: tpm2-sessions: Fix missing tpm_buf_destroy() in tpm2_read_public() tpm2_read_public() calls tpm_buf_init() but fails to call tpm_buf_destroy() on two exit paths, leaking a page allocation: 1. When name_size() returns an error (unrecognized hash algorithm), the function returns directly without destroying the buffer. 2. On the success path, the buffer is never destroyed before returning. All other error paths in the function correctly call tpm_buf_destroy() before returning. Fix both by adding the missing tpm_buf_destroy() calls.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-46147	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: KVM: arm64: Fix pin leak and publication ordering in __pkvm_init_vcpu() Two bugs exist in the vCPU initialisation path: 1. If a check fails after hyp_pin_shared_mem() succeeds, the cleanup path jumps to 'unlock' without calling unpin_host_vcpu() or unpin_host_sve_state(), permanently leaking pin references on the host vCPU and SVE state pages. Extract a register_hyp_vcpu() helper that performs the checks and the store. When register_hyp_vcpu() returns an error, call unpin_host_vcpu() and unpin_host_sve_state() inline before falling through to the existing 'unlock' label. 2. register_hyp_vcpu() publishes the new vCPU pointer into 'hyp_vm->vcpus[]' with a bare store, allowing a concurrent caller of pkvm_load_hyp_vcpu() to observe a partially initialised vCPU object. Ensure the store uses smp_store_release() and the load uses smp_load_acquire(). While 'vm_table_lock' currently serialises the store and the load, these barriers ensure the reader sees the fully initialised 'hyp_vcpu' object even if there were a lockless path or if the lock's own ordering guarantees were insufficient for nested object initialization.
CVE-2026-46148	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: spi: microchip-core-qspi: control built-in cs manually The coreQSPI IP supports only a single chip select, which is automagically operated by the hardware - set low when the transmit buffer first gets written to and set high when the number of bytes written to the TOTALBYTES field of the FRAMES register have been sent on the bus. Additional devices must use GPIOs for their chip selects. It was reported to me that if there are two devices attached to this QSPI controller that the in-built chip select is set low while linux tries to access the device attached to the GPIO. This went undetected as the boards that connected multiple devices to the SPI controller all exclusively used GPIOs for chip selects, not relying on the built-in chip select at all. It turns out that this was because the built-in chip select, when controlled automagically, is set low when active and high when inactive, thereby ruling out its use for active-high devices or devices that need to transmit with the chip select disabled. Modify the driver so that it controls chip select directly, retaining the behaviour for mem_ops of setting the chip select active for the entire duration of the transfer in the exec_op callback. For regular transfers, implement the set_cs callback for the core to use. As part of this, the existing setup callback, mchp_coreqspi_setup_op(), is removed. Modifying the CLKIDLE field is not safe to do during operation when there are multiple devices, so this code is removed entirely. Setting the MASTER and ENABLE fields is something that can be done once at probe, it doesn't need to be re-run for each device. Instead the new setup callback sets the built-in chip select to its inactive state for active-low devices, as the reset value of the chip select in software controlled mode is low.
CVE-2026-46171	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: riscv: kvm: fix vector context allocation leak When the second kzalloc (host_context.vector.datap) fails in kvm_riscv_vcpu_alloc_vector_context, the first allocation (guest_context.vector.datap) is leaked. Free it before returning.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-46228	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: spi: ch341: fix devres lifetime USB drivers bind to USB interfaces and any device managed resources should have their lifetime tied to the interface rather than parent USB device. This avoids issues like memory leaks when drivers are unbound without their devices being physically disconnected (e.g. on probe deferral or configuration changes). Fix the controller and driver data lifetime so that they are released on driver unbind. Note that this also makes sure that the SPI controller is placed correctly under the USB interface in the device tree.
CVE-2026-46256	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: NFS/localio: prevent direct reclaim recursion into NFS via nfs_writepages LOCALIO is an NFS loopback mount optimization that avoids using the network for READ, WRITE and COMMIT if the NFS client and server are determined to be on the same system. But because LOCALIO is still fundamentally "just NFS loopback mount" it is susceptible to recursion deadlock via direct reclaim, e.g.: NFS LOCALIO down to XFS and then back into NFS via nfs_writepages. Fix LOCALIO's potential for direct reclaim deadlock by ensuring that all its page cache allocations are done from GFP_NOFS context. Thanks to Ben Coddington for pointing out commit ad22c7a043c2 ("xfs: prevent stack overflows from page cache allocation").
CVE-2026-46272	MEDIUM	4.7	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: coresight: tmc-etr: Fix race condition between sysfs and perf mode When trying to run perf and sysfs mode simultaneously, the WARN_ON() in tmc_etr_enable_hw() is triggered sometimes: WARNING: CPU: 42 PID: 3911571 at drivers/hwtracing/coresight/coresight-tmc-etr.c:1060 tmc_etr_enable_hw+0xc0/0xd8 [coresight_tmc] [..snip..] Call trace: tmc_etr_enable_hw+0xc0/0xd8 [coresight_tmc] (P) tmc_enable_etr_sink+0x11c/0x250 [coresight_tmc] (L) tmc_enable_etr_sink+0x11c/0x250 [coresight_tmc] coresight_enable_path+0x1c8/0x218 [coresight] coresight_enable_sysfs+0xa4/0x228 [coresight] enable_source_store+0x58/0xa8 [coresight] dev_attr_store+0x20/0x40 sysfs_kf_write+0x4c/0x68 kernfs_fop_write_iter+0x120/0x1b8 vfs_write+0x2c8/0x388 ksys_write+0x74/0x108 __arm64_sys_write+0x24/0x38 el0_svc_common.constprop.0+0x64/0x148 do_el0_svc+0x24/0x38 el0_svc+0x3c/0x130 el0t_64_sync_handler+0xc8/0xd0 el0t_64_sync+0x1ac/0x1b0 ---[end trace 0000000000000000]--- Since the enablement of sysfs mode is separeted into two critical regions, one for sysfs buffer allocation and another for hardware enablement, it's possible to race with the perf mode. Fix this by double check whether the perf mode's been used before enabling the hardware in sysfs mode. mode: [sysfs mode] [perf mode] tmc_etr_get_sysfs_buffer() spin_lock(&drvdata->spinlock) [sysfs buffer allocation] spin_unlock(&drvdata->spinlock) spin_lock(&drvdata->spinlock) tmc_etr_enable_hw() drvdata->etr_buf = etr_perf->etr_buf spin_unlock(&drvdata->spinlock) spin_lock(&drvdata->spinlock) tmc_etr_enable_hw() WARN_ON(drvdata->etr_buf) // WARN sicne etr_buf initialized at the perf side spin_unlock(&drvdata->spinlock) With this fix, we retain the check for CS_MODE_PERF in get_etr_sysfs_buf. This ensures we verify whether the perf mode's already running before we actually allocate the buffer. Then we can save the time of allocating/freeing the sysfs buffer if race with the perf mode.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-50219	MEDIUM	4.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L	expat 2.6.4	libexpat before 2.8.2 lacks handler call depth tracking for calls to XML_GetBuffer, XML_Parse, XML_ParseBuffer, XML_ParserFree, or XML_ParserReset from within handlers in cases of a policy violation. Thus, a use-after-free can occur,
CVE-2025-71315	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/vkms: Convert to DRM's vblank timer Replace vkms' vblank timer with the DRM implementation. The DRM code is identical in concept, but differs in implementation. Vblank timers are covered in vblank helpers and initializer macros, so remove the corresponding hrtimer in struct vkms_output. The vblank timer calls vkms' custom timeout code via handle_vblank_timeout in struct drm_crtc_helper_funcs.
CVE-2026-46314	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/v3d: Reject empty multisync extension to prevent infinite loop v3d_get_extensions() walks a userspace-provided singly-linked list of ioctl extensions without any bound on the chain length. A local user can craft a self-referential extension (ext->next == &ext) with zero in_sync_count and out_sync_count, which bypasses the existing duplicate- extension guard: if (se->in_sync_count || se->out_sync_count) return -EINVAL; The guard never fires because v3d_get_multisync_post_deps() returns immediately when count is zero, leaving both fields at zero on every iteration. The result is an infinite loop in kernel context, blocking the calling thread and pegging a CPU core indefinitely. Fix this by rejecting a multisync extension where both in_sync_count and out_sync_count are zero in v3d_get_multisync_submit_deps(). An empty multisync carries no synchronization information and serves no useful purpose, so returning -EINVAL for such an extension is the correct defense against this attack vector.
CVE-2026-11822	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:H	sqlite3 3.45.3	SQLite before 3.53.2 contains an out-of-bounds read vulnerability in the FTS5 full-text search extension that allows attackers to crash the process by supplying a crafted database with a malformed FTS5 leaf page. Attackers can set an oversized term length in the leaf page so that fts5LeafSeek() reads past the end of the page buffer while comparing terms during an FTS5 MATCH query, in applications compiled with SQLITE_ENABLE_FTS5.
CVE-2026-56131	MEDIUM	4.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L	expat 2.6.4	libexpat before 2.8.2 lacks handler call depth tracking for calls to XML_ResumeParser from within handlers in cases of a policy violation. Thus, a use-after-free can occur (similar to the CVE-2026-50219 situation).
CVE-2026-56132	MEDIUM	6.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L	expat 2.6.4	In libexpat before 2.8.2, there is a heap-based buffer overflow in doProlog in xmlparse.c because scaffold backing array reallocation is mishandled when there is data-structure sharing across parsers.
CVE-2026-56403	MEDIUM	6.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L	expat 2.6.4	libexpat before 2.8.2 has an integer overflow in storeAtts.
CVE-2026-56404	MEDIUM	6.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L	expat 2.6.4	libexpat before 2.8.2 has an integer overflow in addBinding.
CVE-2026-56405	MEDIUM	6.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L	expat 2.6.4	libexpat before 2.8.2 has an integer overflow in getAttributeId.
CVE-2026-56406	MEDIUM	6.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L	expat 2.6.4	libexpat before 2.8.2 has an integer overflow in XML_ParseBuffer because it lacked a check that was present in XML_Parse.
CVE-2026-56407	MEDIUM	6.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L	expat 2.6.4	libexpat before 2.8.2 has an integer overflow in doProlog that is related to storeEntityValue and entity textLen.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-56408	MEDIUM	6.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L	expat 2.6.4	libexpat before 2.8.2 has an integer overflow in copyString.
CVE-2026-56409	MEDIUM	6.5	CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:L	expat 2.6.4	xmlwf in libexpat before 2.8.2 has an integer overflow for the output filename when -d outputDir is used.
CVE-2026-56410	MEDIUM	6.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L	expat 2.6.4	xmlwf in libexpat before 2.8.2 has an integer overflow in resolveSystemId.
CVE-2026-56411	MEDIUM	6.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L	expat 2.6.4	xmlwf in libexpat before 2.8.2 has an integer overflow in endDoctypeDecl via NOTATION declarations.
CVE-2026-56412	MEDIUM	4.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L	expat 2.6.4	libexpat before 2.8.2 does not consider XML_TOK_DATA_CHARS in doCdataSection and thus lacks handler call depth tracking for various calls from within handlers in cases of a policy violation. Thus, a use-after-free can occur. NOTE: this issue exists because of an incomplete fix for CVE-2026-50219.
CVE-2026-12725	MEDIUM	5.9	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H	dnsmasq 2.90	A heap-based buffer overflow was found in dnsmasq. When DNSSEC validation and query logging are both enabled, logging of DS or DNSKEY replies containing unsupported algorithm or digest types can cause dnsmasq to write past the end of an internal logging buffer. A remote attacker able to supply such a DNS response may crash the dnsmasq process, resulting in denial of service.
CVE-2026-12969	MEDIUM	5.3	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	dnsmasq 2.90	An out-of-bounds read vulnerability exists in dnsmasq's find_soa() function in src/rfc1035.c. When parsing NS section records, extract_name() is called with extrabytes=0, failing to validate that 10 additional bytes exist for fixed-length DNS record fields. A remote attacker controlling a DNS zone can exploit this via a crafted NXDOMAIN response to cause a 10-byte heap out-of-bounds read, potentially accessing stale data from prior transactions.
CVE-2026-52937	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: tap: fix stack info leak in tap_ioctl() SIOCGIFHWADDR In the SIOCGIFHWADDR path, tap_ioctl() copies 16 bytes of an uninitialised on-stack struct sockaddr_storage to userspace via ifr_hwaddr, but netif_get_mac_address() only writes sa_family and dev->addr_len (6 for Ethernet) bytes, leaving sa_data[6..13] uninitialised. Those 8 trailing bytes leak kernel stack contents; SIOCGIFHWADDR on a macvtap chardev returns kernel .text and direct-map pointers, defeating KASLR. Initialise ss at declaration.
CVE-2026-52944	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ksmbd: fix FSCTL permission bypass by adding a permission check for FSCTL_SET_SPARSE FSCTL_SET_SPARSE in fsctl_set_sparse() modifies the file's sparse attribute and saves it through xattr without any permission checks. This exposes two issues: 1) A client on a read-only share can change the sparse attribute on files it opened, even though the share is read-only. Other FSCTL write operations already check test_tree_conn_flag(work->tcon, KSMBD_TREE_CONN_FLAG_WRITABLE), but FSCTL_SET_SPARSE does not. 2) Even on writable shares, clients without FILE_WRITE_DATA or FILE_WRITE_ATTRIBUTES access should not modify the sparse attribute. Similar handle-level checks exist in other functions but are missing here. Add both share-level writable check and per-handle access check. Use goto out on error to avoid leaking file references.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-52994	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: vsock/virtio: fix MSG_ZEROCOPY pinned-pages accounting virtio_transport_init_zcopy_skb() uses iter->count as the size argument for msg_zerocopy_realloc(), which in turn passes it to mm_account_pinned_pages() for RLIMIT_MEMLOCK accounting. However, this function is called after virtio_transport_fill_skb() has already consumed the iterator via __zerocopy_sg_from_iter(), so on the last skb, iter->count will be 0, skipping the RLIMIT_MEMLOCK enforcement. Pass pkt_len (the total bytes being sent) as an explicit parameter to virtio_transport_init_zcopy_skb() instead of reading the already-consumed iter->count. This matches TCP and UDP, which both call msg_zerocopy_realloc() with the original message size.
CVE-2026-53018	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: f2fs: avoid reading already updated pages during GC We found the following issue during fuzz testing: page: refcount:3 mapcount:0 mapping:00000000b6e89c65 index:0x18b2dc pfn:0x161ba9 memcg:f8ffff800e269c00 aops:f2fs_meta_aops ino:2 flags: 0x52880000000080a9(locked|waiters|uptodate|lru|private|zone=1|kasantag=0x4a) raw: 52880000000080a9 ffffffec6e17588 ffffffec0ccc088 a7ffff8067063618 raw: 000000000018b2dc 0000000000000009 00000003ffffffff f8ffff800e269c00 page dumped because: VM_BUG_ON_FOLIO(folio_test_uptodate(folio)) page_owner tracks the page as allocated post_alloc_hook+0x58c/0x5ec prep_new_page+0x34/0x284 get_page_from_freelist+0x2dcc/0x2e8c __alloc_pages_noprof+0x280/0x76c __folio_alloc_noprof+0x18/0xac __filemap_get_folio+0x6bc/0xdc4 pagecache_get_page+0x3c/0x104 do_garbage_collect+0x5c78/0x77a4 f2fs_gc+0xd74/0x25f0 gc_thread_func+0xb28/0x2930 kthread+0x464/0x5d8 ret_from_fork+0x10/0x20 ---[cut here]--- kernel BUG at mm/filemap.c:1563! folio_end_read+0x140/0x168 f2fs_finish_read_bio+0x5c4/0xb80 f2fs_read_end_io+0x64c/0x708 bio_endio+0x85c/0x8c0 blk_update_request+0x690/0x127c scsi_end_request+0x9c/0xb8c scsi_io_completion+0xf0/0x250 scsi_finish_command+0x430/0x45c scsi_complete+0x178/0x6d4 blk_mq_complete_request+0xcc/0x104 scsi_done_internal+0x214/0x454 scsi_done+0x24/0x34 which is similar to the problem reported by syzbot: https://syzkaller.appspot.com/bug?extid=3686758660f980b402dc This case is consistent with the description in commit 9bf1a3f ("f2fs: avoid GC causing encrypted file corrupted"): Page 1 is moved from blkaddr A to blkaddr B by move_data_block, and after being written it is marked as uptodate. Then, Page 1 is moved from blkaddr B to blkaddr C, VM_BUG_ON_FOLIO was triggered in the endio initiated by ra_data_block. There is no need to read Page 1 again from blkaddr B, since it has already been updated. Therefore, avoid initiating I/O in this case.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-53103	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: mt76: mt7925: fix potential deadlock in mt7925_roc_abort_sync roc_abort_sync() can deadlock with roc_work(). roc_work() holds dev->mt76.mutex, while cancel_work_sync() waits for roc_work() to finish. If the caller already owns the same mutex, both sides block and no progress is possible. This deadlock can occur during station removal when mt76_sta_state() -> mt76_sta_remove() -> mt7925_mac_sta_remove_link() -> mt7925_mac_link_sta_remove() -> mt7925_roc_abort_sync() invokes cancel_work_sync() while roc_work() is still running and holding dev->mt76.mutex. This avoids the mutex deadlock and preserves exactly-once work ownership.
CVE-2026-53104	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: mt76: Fix memory leak destroying device All MT76 rx queues have an associated page_pool even if the queue is not associated to a NAPI (e.g. WED RRO queues with WED enabled). Destroy the page_pool running mt76_dma_cleanup routine during module unload. Moreover returns pages to the page pool if WED is not enabled for WED RRO queues.
CVE-2026-53107	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: libertas: don't kill URBs in interrupt context Serialization for the TX path was enforced by calling usb_kill_urb()/usb_kill_anchored_urbs(), to prevent transmission before a previous URB was completed. usb_tx_block() can be called from interrupt context (e.g. in the HCD giveback path), so we can't always use it to kill in-flight URBs. Prevent sleeping during interrupt context by checking the tx_submitted anchor for existing URBs. We now return -EBUSY, to indicate there's a pending request.
CVE-2026-53167	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: fuse: limit FUSE_NOTIFY_RETRIEVE to uptodate folios FUSE_NOTIFY_RETRIEVE must be limited to uptodate folios; !uptodate folios can contain uninitialized data. Since FUSE_NOTIFY_RETRIEVE is intended to only return data that is already in the page cache and not wait for data from the FUSE daemon, treat !uptodate folios as if they weren't present. This only has security impact on systems that don't enable automatic zero-initialization of all page allocations via CONFIG_INIT_ON_ALLOC_DEFAULT_ON or init_on_alloc=1.
CVE-2026-53226	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: gpio: rockchip: fix generic IRQ chip leak on remove The driver allocates domain generic chips using irq_alloc_domain_generic_chips() during probe. However, on driver remove/teardown, the generic chips are not automatically freed when the IRQ domain is removed because the domain flags do not include IRQ_DOMAIN_FLAG_DESTROY_GC. This causes both the domain generic chips structure and the associated generic chips to be leaked. Additionally, the generic chips remain on the global gc_list and may later be visited by generic IRQ chip suspend, resume, or shutdown callbacks after the GPIO bank has been removed, potentially resulting in a use-after-free and kernel crash. Fix the resource leak by explicitly calling irq_domain_remove_generic_chips() before removing the IRQ domain in rockchip_gpio_remove().

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-53258	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: fix leak if split 6 GHz scanning fails rdev->int_scan_req is leaked if cfg80211_scan() fails. Note that it's supposed to be released at ___cfg80211_scan_done() but this doesn't happen as rdev->scan_req is NULL at that point, too, leading to the early return from the freeing function. unreferenced object 0xffff8881161d0800 (size 512): comm "wpa_supplicant", pid 379, jiffies 4294749765 hex dump (first 32 bytes): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... 00 00 00 00 00 00 00 00 f0 81 13 16 81 88 ff ff ..... backtrace (crc c867fdb6): kmemleak_alloc+0x89/0x90 __kmalloc_noprof+0x2fd/0x410 cfg80211_scan+0x133/0x730 nl80211_trigger_scan+0xc69/0x1cc0 genl_family_rcv_msg_doit+0x204/0x2f0 genl_rcv_msg+0x431/0x6b0 netlink_rcv_skb+0x143/0x3f0 genl_rcv+0x27/0x40 netlink_unicast+0x4f6/0x820 netlink_sendmsg+0x797/0xce0 __sock_sendmsg+0xc4/0x160 ____sys_sendmsg+0x5e4/0x890 ___sys_sendmsg+0xf8/0x180 __sys_sendmsg+0x136/0x1e0 __x64_sys_sendmsg+0x76/0xc0 x64_sys_call+0x13f0/0x17d0 Found by Linux Verification Center (linuxtesting.org).
CVE-2026-53297	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: mana: Guard mana_remove against double invocation If PM resume fails (e.g., mana_attach() returns an error), mana_probe() calls mana_remove(), which tears down the device and sets gd->gdma_context = NULL and gd->driver_data = NULL. However, a failed resume callback does not automatically unbind the driver. When the device is eventually unbound, mana_remove() is invoked a second time. Without a NULL check, it dereferences gc->dev with gc == NULL, causing a kernel panic. Add an early return if gdma_context or driver_data is NULL so the second invocation is harmless. Move the dev = gc->dev assignment after the guard so it cannot dereference NULL.
CVE-2026-53321	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: io_uring/napi: cap busy_poll_to 10 msec Currently there's no cap on the maximum amount of time that napi is allowed to poll if no events are found, which can lead to kernel complaints on a task being stuck as there's no conditional rescheduling done within that loop. Just cap it to 10 msec in total, that's already way above any kind of sane value that will reap any benefits, yet low enough that it's nowhere near being able to trigger preemption complaints.
CVE-2026-13595	MEDIUM	6.8	CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:H	util-linux 2.39.3	A flaw was found in the libblkid library of util-linux. During nested partition probing, the BSD, Minix, Solaris x86, and UnixWare partition probers cache a raw pointer to a parent partition entry in a dynamically allocated array. When subsequent partition additions cause the array to be reallocated, this pointer becomes stale, leading to a heap use-after-free read. An attacker who can present a crafted block device image (for example, via USB insertion or a loop-mounted disk image) can trigger this flaw without user interaction, as libblkid is invoked automatically by udev/udisks as root on block-device hot-plug events. This could lead to limited information disclosure or denial of service.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-58010	MEDIUM	6.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:L	glib-2.0 2.78.6	A flaw was found in GLib. An off-by-one error can occur in the gvs_tuple_is_normal function in the glib/gvariant-serialiser.c file when doing an alignment padding check because the bounds check uses > instead of >=, causing an out-of-bounds read of only 1 byte. This issue can cause a minor information disclosure of 1 byte and a denial of service when the out-of-bounds read crosses a page boundary.
CVE-2026-58011	MEDIUM	6.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L	glib-2.0 2.78.6	A flaw was found in GLib. An out-of-bounds read of only 2 bytes can occur in the g_date_time_get_ymd function in the glib/gdatetime.c file when an invalid GDateTime object produced by the g_date_time_add_full function is processed. This flaw can corrupt the date output and potentially cause logic errors that may lead to a denial of service.
CVE-2026-58012	MEDIUM	6.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:L	glib-2.0 2.78.6	A flaw was found in GLib. A buffer over-read can occur in the g_regex_replace function when used with the `G_REGEX_RAW` compile flag and case-change replacement escapes because the string_append function processes matched substrings using UTF-8 functions that assume valid UTF-8 input, even when the string is treated as raw bytes. This vulnerability can cause a minor information disclosure of 1-5 bytes and a denial of service when the buffer over-read crosses a page boundary.
CVE-2026-58013	MEDIUM	6.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:L	glib-2.0 2.78.6	A flaw was found in GLib. A buffer over-read can occur in g_io_channel_read_line_backend() in the giochannel.c file when a custom line terminator with a length greater than one is set, causing memcmp to read past the GString buffer. This vulnerability can cause a minor information disclosure of 7 bytes or a denial of service when the buffer over-read crosses a page boundary.
CVE-2026-58015	MEDIUM	5.9	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N	glib-2.0 2.78.6	A flaw was found in GLib. The D-Bus client-side implementation of the DBUS_COOKIE_SHA1 SASL authentication mechanism does not validate the cookie_context parameter received from the server. A malicious D-Bus server can supply a cookie_context containing path traversal sequences, causing the client to read an arbitrary file and exfiltrate sensitive data by verifying guessed file contents against a generated hash.
CVE-2026-8458	MEDIUM	6.5	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:N	curl 8.7.1	libcurl might in some circumstances reuse the wrong connection when asked to do Negotiate-authenticated ones, even when they are set to use different "services". libcurl features a pool of recent connections so that subsequent requests can reuse an existing connection to avoid overhead. When reusing a connection a range of criteria must be met. Due to a logical error in the code, a request that was issued by an application could wrongfully reuse an existing connection to the same server that was authenticated using different services.
CVE-2026-59995	MEDIUM	4.2	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:L	openssh 9.6p1	sftp in OpenSSH before 10.4 does not properly constrain the location of downloaded files when "sftp server:/path ." is used with an attacker-controlled server.
CVE-2026-59996	MEDIUM	4.2	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:L	openssh 9.6p1	scp in OpenSSH before 10.4 may place a file in the parent directory of an intended directory when the copy occurs between two remote destinations.
CVE-2026-59997	MEDIUM	4.2	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N	openssh 9.6p1	internal-sftp in sshd in OpenSSH before 10.4 recognizes only the first 9 command-line arguments, which can be important if a later command-line argument would have helped to ensure the intended security properties of an SFTP connection.
CVE-2026-59998	MEDIUM	4.8	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N	openssh 9.6p1	sshd in OpenSSH before 10.4 has an undocumented security-relevant behavior: GSSAPIStrictAcceptorCheck has no value if the server is in Windows Active Directory.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-59999	MEDIUM	5.9	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:H/A:N	openssh 9.6p1	In sshd in OpenSSH before 10.4, DisableForwarding=yes was supposed to take precedence over PermitTunnel=yes, but did not.
CVE-2026-60001	MEDIUM	6.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L	openssh 9.6p1	sshd in OpenSSH before 10.4 does not always honor the minimum authentication delay.
CVE-2026-53365	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: vsock/virtio: fix zerocopy completion for multi-skb sends When a large message is fragmented into multiple skbs, the zerocopy uarg is only allocated and attached to the last skb in the loop. Non-final skbs carry pinned user pages with no completion tracking, so the kernel has no way to notify userspace when those pages are safe to reuse. If the loop breaks early the uarg is never allocated at all, leaking pinned pages with no completion notification. Fix this by following the approach used by TCP: allocate the zerocopy uarg (if not provided by the caller) before the send loop and attach it to every skb via skb_zcopy_set(), which takes a reference per skb. Each skb's completion properly decrements the refcount, and the notification only fires after the last skb is freed. On failure, if no data was sent, the uarg is cleanly aborted via net_zcopy_put_abort(). This issue was initially discovered by sashiko while reviewing commit 1cb36e252211 ("vsock/virtio: fix MSG_ZEROCOPY pinned-pages accounting") but was pre-existing.
CVE-2026-53377	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/msm: always recover the gpu Previously, in case there was no more work to do, recover worker wouldn't trigger recovery and would instead rely on the gpu going to sleep and then resuming when more work is submitted. Recover_worker will first increment the fence of the hung ring so, if there's only one job submitted to a ring and that causes an hang, it will early out. There's no guarantee that the gpu will suspend and resume before more work is submitted and if the gpu is in a hung state it will stay in that state and probably trigger a timeout again. Just stop checking and always recover the gpu. Patchwork: https://patchwork.freedesktop.org/patch/704066/

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-63811	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: f2fs: read COW data with the original inode during atomic write When updating an atomic-write file, f2fs_write_begin() may read the previously written data back from the COW inode: prepare_atomic_write_begin() locates the block in the COW inode and sets use_cow, and the read bio is then built with the COW inode: f2fs_submit_page_read(use_cow ? F2FS_I(inode)->cow_inode : inode, ...); and f2fs_grab_read_bio() decides whether to schedule fs-layer decryption (STEP_DECRYPT) for the bio based on that inode via fscrypt_inode_uses_fs_layer_crypto(). However, the folio being filled belongs to the original inode (folio->mapping->host == inode), and the data stored in the COW block was encrypted (or left as plaintext) using the original inode's context, not the COW inode's -- see f2fs_encrypt_one_page(), which keys off fio->page->mapping->host. fscrypt_decrypt_pagecache_blocks() likewise operates on folio->mapping->host. The COW inode is created as a tmpfile in the parent directory and inherits its encryption policy from there. With test_dummy_encryption the newly created COW inode gets the dummy policy and becomes encrypted, while a pre-existing regular file -- created before the policy applied, e.g. already present in the on-disk image -- stays unencrypted. The read path then sets STEP_DECRYPT based on the encrypted COW inode and calls fscrypt_decrypt_pagecache_blocks() on a folio whose host (the unencrypted original inode) has a NULL ->i_crypt_info, dereferencing it: Oops: general protection fault, probably for non-canonical address ... KASAN: null-ptr-deref in range [0x0000000000000008-0x000000000000000f] RIP: 0010:fscrypt_decrypt_pagecache_blocks+0xa0/0x310 Workqueue: f2fs_post_read_wq f2fs_post_read_work Call Trace: fscrypt_decrypt_bio+0x1eb/0x340 f2fs_post_read_work+0xba/0x140 process_one_work+0x91c/0x1a40 worker_thread+0x677/0xe90 kthread+0x2bc/0x3a0 The COW inode is only needed to locate the on-disk block, and that block address is already resolved into @blkaddr by prepare_atomic_write_begin() via __find_data_block(cow_inode, ...); f2fs_submit_page_read() then reads from that physical @blkaddr directly, so the inode argument only selects the post-read crypto context, not which block is fetched. Reading with @inode therefore returns the same (latest, not-yet-committed) COW data, while making both the fs-layer decryption decision and the inline crypto path use the correct (original inode's) key. With the COW inode no longer used at the read site, the use_cow flag has no remaining consumer; drop it from f2fs_write_begin() and prepare_atomic_write_begin().
CVE-2026-63859	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: airoha: Add missing bits in airoha_qdma_cleanup_tx_queue() Similar to airoha_qdma_cleanup_rx_queue(), reset DMA TX descriptors in airoha_qdma_cleanup_tx_queue routine. Moreover, reset TX_DMA_IDX to TX_CPU_IDX to notify the NIC the QDMA TX ring is empty.
CVE-2026-63998	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ethtool: module: call ethnl_ops_complete() on module flash errors When validate() fails we are skipping over ethnl_ops_complete() even tho we already called ethnl_ops_begin().

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-63999	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ethtool: rss: fix indir_table and hkey leak on get_rxfh failure rss_prepare_get() allocates the indirection table and hash key buffer via rss_get_data_alloc(), then calls ops->get_rxfh() to populate them. If get_rxfh() fails, the function returns an error without freeing the allocation.
CVE-2026-64006	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: nf_tables: fix dst corruption in same register operation For lshift and rshift, the shift operations are performed in a loop over 32-bit words. The loop calculates the shifted value and write it to dst, and then immediately reads from src to calculate the carry for the next iteration. Because src and dst could point to the same memory location, the carry is incorrectly calculated using the newly modified dst value instead of the original src value. Adding a temporary local variable to cache the original value before writing to dst and using it for the carry calculation solves the problem. In addition, partial overlap is rejected from control plane for all kind of operations including byteorder. This was tested with the following bytecode: table test_table ip flags 0 use 1 handle 1 ip test_table test_chain use 3 type filter hook input prio 0 policy accept packets 0 bytes 0 flags 1 ip test_table test_chain 2 [immediate reg 1 0x44332211 0x88776655] [bitwise reg 1 = (reg 1 << 0x08000000)] [cmp eq reg 1 0x66443322 0x00887766] [counter pkts 0 bytes 0] ip test_table test_chain 4 3 [immediate reg 1 0x44332211 0x88776655] [bitwise reg 1 = (reg 1 << 0x08000000)] [cmp eq reg 1 0x55443322 0x00887766] [counter pkts 21794 bytes 1917798]
CVE-2026-64038	MEDIUM	4.7	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: hwmon: (lm90) Stop work before releasing hwmon device Sashiko reports: In lm90_probe(), the devm action to cancel the alert_work and report_work (lm90_restore_conf) is registered in lm90_init_client() before devm_hwmon_device_register_with_info() is called. Because devm executes cleanup actions in reverse order during module unbind or probe failure, the hwmon device is unregistered and freed first. If lm90_alert_work() or lm90_report_alarms() runs in the window between the hwmon device being freed and the delayed works being cancelled, lm90_update_alarms() will dereference the freed data->hwmon_dev here. Fix the problem by canceling the workers separately after registering the hwmon device and before registering the interrupt handler. This ensures that the workers are canceled after interrupts are disabled and before the hwmon device is released. Add "shutdown" flag to indicate that device shutdown is in progress to prevent workers from being re-armed.
CVE-2026-64060	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfs: Fix leak of request in netfs_write_begin() error handling Fix netfs_write_begin() to not leak our ref on the request in the event that we get an error from netfs_wait_for_read().

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-64070	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: powerpc/hv-gpci: fix preempt count leak in sysfs show paths Four sysfs show() callbacks in hv-gpci take get_cpu_var(hv_gpci_reqb) (which calls preempt_disable()) but only call the matching put_cpu_var() on the error path under the 'out:' label. Every successful read leaks one preempt_disable(): processor_bus_topology_show() processor_config_show() affinity_domain_via_virtual_processor_show() affinity_domain_via_domain_show() (affinity_domain_via_partition_show() was already correct.) On a CONFIG_PREEMPT=y kernel, repeated reads raise preempt_count and eventually return to userspace with preemption still disabled. The next user-mode page fault then hits faulthandler_disabled() == 1, gets forced to SIGSEGV, and the resulting coredump trips 'BUG: scheduling while atomic' in call_usermodehelper_exec -> wait_for_completion_state -> schedule: BUG: scheduling while atomic: //0x00000004 ... __schedule_bug+0x6c/0x90 __schedule+0x58c/0x13a0 schedule+0x48/0x1a0 schedule_timeout+0x104/0x170 wait_for_completion_state+0x16c/0x330 call_usermodehelper_exec+0x254/0x2d0 vfs_coredump+0x1050/0x2590 get_signal+0xb9c/0xc80 do_notify_resume+0xf8/0x470 Add an out_success label that calls put_cpu_var() before returning the byte count, mirroring affinity_domain_via_partition_show().
CVE-2026-64190	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: team: fix NULL pointer dereference in team_xmit during mode change __team_change_mode() clears team->ops with memset() before restoring safe dummy handlers via team_adjust_ops(). A concurrent team_xmit() running under RCU on another CPU can read team->ops.transmit during this window and call a NULL function pointer, crashing the kernel. The race requires a mode change (CAP_NET_ADMIN) concurrent with transmit on the team device. BUG: kernel NULL pointer dereference, address: 0000000000000000 Oops: 0010 [#1] SMP KASAN NOPTI RIP: 0010:0x0 Call Trace: team_xmit (drivers/net/team/team_core.c:1853) dev_hard_start_xmit (net/core/dev.c:3904) __dev_queue_xmit (net/core/dev.c:4871) packet_sendmsg (net/packet/af_packet.c:3109) __sys_sendto (net/socket.c:2265) The original code assumed that no ports means no traffic, so mode changes could freely memset()/memcpy() the ops. AF_PACKET with forced carrier breaks that assumption. Prevent the race instead of making it safe: replace memset()/memcpy() with per-field updates that never touch transmit or receive. Those two handlers are managed solely by team_adjust_ops(), which already installs dummies when tx_en_port_count == 0 (always true during mode change since no ports are present). WRITE_ONCE/READ_ONCE prevent store/load tearing on the handler pointers. synchronize_net() before exit_op() drains in-flight readers that may still reference old mode state from before port removal switched the handlers to dummies.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-64205	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: i2c: i801: fix hardware state machine corruption in error path A severe livelock and subsequent Hung Task panic were observed in the i2c-i801 driver during concurrent Fuzzing. The crash is caused by an unconditional hardware register cleanup in the error handling path of i801_access(). When i801_check_pre() fails (e.g., returning -EBUSY because the SMBus controller is actively used by BIOS/ACPI), the kernel does not actually acquire the hardware ownership. However, the code jumps to the 'out' label and executes: iowrite8(SMBHSTSTS_INUSE_STS | STATUS_FLAGS, SMBHSTSTS(priv)); This forcefully clears the INUSE_STS lock and resets the hardware status flags without owning the controller. Doing so interrupts ongoing BIOS/ACPI transactions and totally corrupts the SMBus hardware state machine. Consequently, all subsequent i801_access() calls fail at the pre-check stage, triggering an endless stream of "SMBus is busy, can't use it!" error logs. Over a slow serial console, this printk flood monopolizes the CPU (Console Livelock), starving other processes trying to acquire the mmap_lock down_read semaphore, ultimately triggering the hung task watchdog. Fix this by moving the 'out' label below the hardware register cleanup. If i801_check_pre() fails, we safely bypass the iowrite8() and only release the software locks (pm_runtime and mutex), strictly adhering to the rule of not releasing resources that were never acquired.
CVE-2026-64212	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: iwlwifi: mld: don't dereference a pointer before NULL checking it In iwl_mld_remove_link, the link->fw_id is saved at the beginning of the function so we have it after we freed the link. But the link pointer can be NULL, and is not checked when the fw_id is stored. Fix it by simply freeing the link at the end of the function. fFixes: 0e66a39f4f0e ("wifi: iwlwifi: fix potential use after free in iwl_mld_remove_link()")
CVE-2026-64213	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: hwmon: (lm90) Add lock protection to lm90_alert Sashiko reports: lm90_alert() executes in the smbus alert context and calls lm90_update_confreg() to disable the hardware alert line, without acquiring hwmon_lock. Concurrently, sysfs write operations (such as lm90_write_convrate) hold the hwmon_lock, temporarily modify data->config, and then restore it. If an alert interrupt occurs concurrently with a sysfs write, the sysfs path will overwrite the alert handler's modifications to data->config and the hardware register. This unintentionally re-enables the hardware alert line while the alarm is still active, causing an interrupt storm. Add the missing lock to lm90_alert() to solve the problem.
CVE-2026-64290	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: iommufd: Break the loop on failure in iommufd_fault_fops_read() On a copy_to_user() failure inside the inner list_for_each_entry, only the inner loop breaks; the outer while re-fetches the just-restored fault group and retries the failing copy_to_user() forever, spinning the reader at 100% CPU with fault->mutex held. Check rc after the inner loop and break the outer while as well.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-64513	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: KVM: x86: Unconditionally recompute CR8 intercept on PPR update The TPR_THRESHOLD field in the VMCS is used by VMX to induce VM exits when the guest's virtual TPR falls under the specified threshold, allowing KVM to inject previously masked interrupts. KVM handles these VM exits in handle_tpr_below_threshold(). Commit eb90f3417a0c ("KVM: vmx: speed up TPR below threshold vmexits") optimized this function by calling apic_update_ppr() instead of raising KVM_REQ_EVENT. apic_update_ppr() then raises KVM_REQ_EVENT if there is a pending, deliverable interrupt. However, if there are no new interrupts pending, apic_update_ppr() does not issue the request. Thus, kvm_lapic_update_cr8_intercept() and vmx_update_cr8_intercept() are not called before VM entry, which results in a high, stale TPR_THRESHOLD. This is problematic due to the following sentence in 28.2.1.1 "VM-Execution Control Fields" in the SDM: The following check is performed if the “use TPR shadow” VM-execution control is 1 and the “virtualize APIC accesses” and “virtual-interrupt delivery” VM-execution controls are both 0: the value of bits 3:0 of the TPR threshold VM-execution control field should not be greater than the value of bits 7:4 of VTPR. This error condition is typically not observed when KVM runs on a bare metal system because modern processors support APICv, which enables virtual-interrupt delivery, and which KVM uses when possible. This causes the processor to no longer generate TPR-below-threshold exits and to no longer check TPR_THRESHOLD on entry. However, when running on older platforms, or under nested virtualization on a hypervisor that does not support virtual-interrupt delivery and enforces this check (like Hyper-V) this can cause a VM entry failure with hardware error 0x7, as seen in [1]. Call kvm_lapic_update_cr8_intercept() if apic_update_ppr() does not find a deliverable interrupt (and thus does not raise KVM_REQ_EVENT). Remove calls to kvm_lapic_update_cr8_intercept() on paths that end up in apic_update_ppr(), as they now become redundant. This ensures that any path that updates the guest's PPR also figures out if KVM needs to wait for a TPR change (using TPR_THRESHOLD on VMX or CR8 intercepts on SVM).

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-64569	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: mpls: fix NULL deref in mpls_valid_fib_dump_req() on CONFIG_INET=n On CONFIG_INET=n builds, mpls_valid_fib_dump_req() walks the parsed attribute table itself instead of calling ip_valid_fib_dump_req(). The RTA_OIF arm passes tb[RTA_OIF] to nla_get_u32() without checking it is present, so an RTM_GETROUTE dump for AF_MPLS with strict checking and no RTA_OIF hits a NULL dereference. RTM_GETROUTE is RTNL_KIND_GET, which rtnetlink_rcv_msg() permits without CAP_NET_ADMIN, so an unprivileged user can trigger it. Oops: general protection fault, probably for non-canonical address 0xdfffffc0000000000: 0000 [#1] SMP KASAN NOPTI KASAN: null-ptr-deref in range [0x0000000000000000-0x0000000000000007] RIP: 0010:mpls_valid_fib_dump_req (net/mpls/af_mpls.c:2189) Call Trace: mpls_dump_routes (net/mpls/af_mpls.c:2236) netlink_dump (net/netlink/af_netlink.c:2331) __netlink_dump_start (net/netlink/af_netlink.c:2446) rtnetlink_rcv_msg (net/core/rtnetlink.c:7033) netlink_rcv_skb (net/netlink/af_netlink.c:2556) netlink_unicast (net/netlink/af_netlink.c:1345) netlink_sendmsg (net/netlink/af_netlink.c:1900) __sock_sendmsg (net/socket.c:790) ____sys_sendmsg (net/socket.c:2684) ___sys_sendmsg (net/socket.c:2738) __sys_sendmsg (net/socket.c:2770) do_syscall_64 (arch/x86/entry/syscall_64.c:94) entry_SYSCALL_64_after_hwframe (arch/x86/entry/entry_64.S:121) Skip unset attributes, as ip_valid_fib_dump_req() does.
CVE-2026-63074	MEDIUM	5.9	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H	openssl 3.5.7	Issue summary: The OpenSSL Certificate Management Protocol (CMP) caches additional certificates (extraCerts) sent in a CMP message, but never expunges them (for instance if they are invalid). If a server reuses an OSSL_CMP_CTX frequently, this cache of extraCerts may grow unboundedly, and a malicious client may flood a CMP server with requests driving this growth. Impact summary: Users utilizing a CMP server that reuses a single OSSL_CMP_CTX for the lifetime of a server process may observe unbounded memory growth in the event a malicious client repeatedly sends requests containing unique extra certificates, which may lead to OOM conditions. CWE: CWE-770: Allocation of Resources Without Limits or Throttling Description: If a remote user sends CMP messages to a server with a list of extraCerts and the message is rejected, the extraCerts from the message remains in the server contexts untrusted certificate stack. This exposes servers with long lived ctx objects to Denial of Service attacks in which an attacker sends messages intending to be rejected with a large list of additional certificates repeatedly, forcing the server to store them indefinitely. The issue was fixed by removing the added extra certs if the message is rejected, using the same method as when the context is configured to not do caching at all. FIPS impact: no As the CMP code lives outside the FIPS module boundary, no FIPS modules are affected by this CVE.
CVE-2026-98051	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: bcmasp: fix tx_spb_ring_full() checking same slot cnt times The loop initialised next_index from intf->tx_spb_index on every iteration, so incr_ring() always produced the same result and only one slot was ever tested. Move the initialisation before the loop so each iteration advances next_index and the function correctly checks that cnt consecutive descriptor slots are available before allowing a new transmission.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98053	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ASoC: Intel: avs: Refactor and fix init_config access Existing code accesses enties found in ->init_configs array through indexes that are part of ->config_ids array. Those two are limited by: ->num_init_configs and ->num_config_ids respectively. Using ID larger or equal to ->num_init_configs leads to out-of-bounds access: avs_path_module_send_init_configs() loop: (...) &acomp->tplg->init_configs[ids[i]] ^ out-of-bounds candidate Rather than adding another if-statement, refactor the code. There is no need to store the IDs, have a list of pointers to actual config-entries instead. As the verification of ->init_config entries does not differ from verification of other types that are part of the topology.c file, simply reuse the code.
CVE-2026-98054	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ASoC: Intel: avs: Fix unbalanced module reference count strace_open() invokes try_module_get() which on success takes the module reference. If any follow up operation causes strace_open() to fail, the refcount shall be put down.
CVE-2026-98055	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ASoC: Intel: avs: Clean up the bus when fetching ML caps fails snd_hdac_ext_bus_get_ml_capabilities() may fail and its return code shall be checked and accounted for. Address the issue by updating the error-path for avs_pci_probe(). At the same time, if the function in question succeeds but the next part of avs_pci_probe() fails, the hlink list shall be cleaned up before leaving the scope.
CVE-2026-98057	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ring-buffer: Add checking nr_subbufs to persistent ring buffer validation Sashiko reported that the code was using meta->nr_subbufs without making sure that it matched the nr_pages + 1 on data that was assuming the two were the same. Add a check to the persistent ring buffer validation code to make sure that the saved nr_subbufs matches what we expect.
CVE-2026-98058	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: Mark syscall helpers as sleepable bpf_sys_bpf() executes the bpf(2) syscall body, which can take mutexes, allocate with GFP_KERNEL, and wait for an RCU grace period. bpf_sys_close() reaches close_fd() and filp_close(), which can sleep as well. Both helpers are limited to BPF_PROG_TYPE_SYSCALL, whose main program is sleepable. That does not make every callback sleepable: a syscall program can register a bpf_timer callback, and the verifier checks that callback in a non-sleepable context while retaining the syscall helper set. Without .might_sleep on the prototypes, such a callback can invoke bpf_sys_bpf() from hrtimer softirq context and trigger a scheduling-while-atomic failure. bpf_sys_close() is exposed through the same missing context check. Set .might_sleep on both prototypes so the existing helper-context check rejects them from timer callbacks and other atomic regions. Calls from the sleepable main body remain valid.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98059	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: Mark sched_process_wait argument as nullable do_wait() passes wo->wo_pid to the sched_process_wait tracepoint. kernel_wait4() leaves wo_pid NULL for wait4(-1), and kernel_waitid_prepare() does likewise for waitid(P_ALL). btf_ctx_access() currently types argument 0 as PTR_TO_BTF_ID | PTR_TRUSTED. Without PTR_MAYBE_NULL, the verifier accepts an unchecked dereference. Trusted pointer loads have no fault protection, so a wait for any child can then cause a NULL pointer dereference in JITed BPF code. Add sched_process_wait to raw_tp_null_args[] with argument 0 marked nullable. The verifier rejects an unchecked dereference while preserving access after the program checks the pointer for NULL.
CVE-2026-98062	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: Mark signal tracepoint siginfo arguments as scalar The signal_generate and signal_deliver tracepoints declare their info argument as a struct kernel_siginfo pointer. btf_ctx_access() therefore treats it as a trusted pointer for tp_btf programs. Signal delivery also uses SEND_SIG_NOINFO and SEND_SIG_PRIV as special values for this argument. Those values are zero and one respectively, and are not pointers. A tp_btf program can currently dereference either value and fault the kernel. In particular, signal_generate can run from timer interrupt context, turning the fault into a kernel panic. Record both tracepoints in raw_tp_null_args[] and mark argument one as a non-pointer. This preserves scalar access to the cookie while rejecting direct and helper-mediated pointer use. Merely marking it nullable would not suffice because SEND_SIG_PRIV is nonzero.
CVE-2026-98063	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: Fix NULL-ptr-deref in btf_var_show() btf_var_show() calls btf_type_id_resolve() unconditionally, which dereferences btf->resolved_ids. That is NULL for a base BTF - e.g. the vmlinux BTF that bpf_snprintf_btf() renders against - since base BTF is not resolved during parsing. btf_modifier_show() guards this with 'if (btf->resolved_ids)', but btf_var_show() does not. A BPF program that passes the type_id of a BTF_KIND_VAR from the vmlinux BTF to bpf_snprintf_btf() thus NULL-derefs: KASAN: probably user-memory-access in range [0x46638-0x4663f] RIP: 0010:btf_var_show (kernel/bpf/btf.c:2929) Call Trace: btf_type_show (kernel/bpf/btf.c:8259) btf_type_snprintf_show (kernel/bpf/btf.c:8329) bpf_snprintf_btf (kernel/trace/bpf_trace.c:1047) bpf_prog_test_run_raw_tp (net/bpf/test_run.c:829) __sys_bpf (kernel/bpf/syscall.c:4804) do_syscall_64 (arch/x86/entry/syscall_64.c:84) entry_SYSCALL_64_after_hwframe (arch/x86/entry/entry_64.S:121) Resolve the var's type directly with btf_type_skip_modifiers() when resolved_ids is NULL, mirroring btf_modifier_show().

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98064	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: Fix NULL-ptr-deref when showing a void BTF type btf_modifier_show() resolves the modifier and then calls btf_type_ops(t)->show() unconditionally. For the void type (type_id 0, BTF_KIND_UNKN) kind_ops[] has no entry, so ->show is NULL. A "const void" (a modifier resolving to void) cannot be a map key or value - map_check_btf() rejects it because void has no size - so the map dump path does not reach it. But bpf_snprintf_btf() takes a type_id straight from the BPF program, and passing such a "const void" from the vmlinux BTF NULL-derefs: KASAN: null-ptr-deref in range [0x0000000000000028-0x000000000000002f] RIP: 0010:btf_modifier_show (kernel/bpf/btf.c:2914) Call Trace: btf_type_show (kernel/bpf/btf.c:8251) btf_type_snprintf_show (kernel/bpf/btf.c:8321) bpf_snprintf_btf (kernel/trace/bpf_trace.c:1047) bpf_prog_test_run_raw_tp (net/bpf/test_run.c:829) __sys_bpf (kernel/bpf/syscall.c:4804) do_syscall_64 (arch/x86/entry/syscall_64.c:94) entry_SYSCALL_64_after_hwframe (arch/x86/entry/entry_64.S:121) Fall back to btf_df_show() when the resolved type has no show op; it emits the "" placeholder already used for kinds like FWD and FUNC. bpf_snprintf_btf() then returns the length as usual.
CVE-2026-98067	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: erofs: disable LZ4 rolling decompression for now LZ4 rolling decompression [1] was introduced to reduce the memory footprint of temporary pages: For many cases, it is needed for users to read small data within a compressed extent (pcluster), either due to random small read, or since uptodate folios (typically order-0) cannot be reused for decompression again since decompression algorithm refills already-uptodate folios. Rolling decompression works because LZ4 is LZ77-based and only refers to the most recent 64 KiB of decompressed data, so in theory only a bounded rolling window of temporary pages is needed when decompressing. It can save a lot of temporary memory, e.g. 601,960-byte data can be compressed into a 256k LZ4 compressed extent, which means it needs 146 extra pages per request in the worst case if rolling decompression is disabled. However, the upstream LZ4 implementation is not under EROFS' control: For example, the literal copy memmove() may still **copy long literals backward** on x86 based on the address comparison even when the source and destination ranges do not overlap (IOWs, inline decompression doesn't need to be considered here). That breaks the rolling assumption and makes the optimization broken. Disable it for now to make sure the data correctness first since EROFS is used everywhere now: The rolling window approach can be revived once we either ensure that the official LZ4 code always copies forward for non-overlapping ranges or maintain our own LZ4 implementation in EROFS. The main impact is a higher runtime memory footprint; However, recent commit 0f6273ab4637 ("erofs: add a reserved buffer pool for lz4 decompression") helps mitigate this when enabled but it's still not perfect. [1] https://www.usenix.org/conference/atc19/presentation/gao § 3.3 Decompression

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98068	MEDIUM	4.7	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net/rds: don't let rds_conn_shutdown() consume a concurrent drop rds_conn_shutdown() finishes by moving the path from RDS_CONN_DISCONNECTING to RDS_CONN_DOWN, and also accepts RDS_CONN_ERROR as the starting state of that final transition, so that a FIN processed in softirq context during the teardown does not derail the shutdown into a noisy error path. But consuming that RDS_CONN_ERROR also consumes the shutdown pass that came with it: rds_conn_path_drop() sets RDS_CONN_ERROR and then queues cp_down_w, and a pass that starts on a path already in RDS_CONN_DOWN is a no-op. For the FIN case that is harmless - the socket the FIN arrived on is the very socket the teardown just released. It is not harmless for a dropper that attached something to the path first. rds_tcp_accept_one() is such a dropper. Its path claim in rds_tcp_accept_one_path() transitions RDS_CONN_DOWN -> RDS_CONN_CONNECTING, and a concurrent drop - a FIN on a previous socket in softirq context, an administrative reset - can put the path into RDS_CONN_ERROR between that claim and the state check that follows, which accepts RDS_CONN_ERROR. The accept then installs the freshly accepted socket with rds_tcp_set_callbacks() while the queued teardown - which sampled tc->t_sock before this socket existed - is still running. rds_connect_path_complete() fails its transition to RDS_CONN_UP and drops the path again, queueing the pass that should reap the socket it just installed. If the in-flight shutdown's final transition consumes that drop's RDS_CONN_ERROR, the queued pass finds the path in RDS_CONN_DOWN and does nothing. The installed socket is never torn down: it sits established with its callbacks armed and its rds_tcp_connection on rds_tcp_tc_list, the peer sees a connection that nothing ever reads, and the path is wedged in RDS_CONN_DOWN until some later event drops it again. Reproduced with widened race windows as an ever-growing receive queue on a socket owned by a path stuck in RDS_CONN_DOWN, with the peer's send path wedged behind it. Make the final transition only DISCONNECTING -> DOWN. If it fails because the path is in RDS_CONN_ERROR, a drop raced the teardown: cancel the reconnect timer and clear RDS_RECONNECT_PENDING - the one piece of the skipped tail that must not be left behind - and return, letting the pass the drop queued finish the job: it tears down whatever attached to the path in the meantime, completes the transition to RDS_CONN_DOWN, and re-arms the reconnect from its own tail. The timer quiesce in that branch matters because the racing drop does not always queue that pass: rds_conn_path_drop() returns without queueing when a destroy is pending - exactly the situation during a netns teardown or module unload, when a FIN on the dying socket is processed while rds_conn_path_destroy() flushes cp_down_w. If the flushed pass is the one that takes this return, no later pass exists, and rds_conn_path_destroy() would find cp_conn_w still armed (WARN_ON) and then free a path whose reconnect timer can still fire. With the cancel in the branch, every exit of a shutdown pass leaves the timer quiesced no matter which pass completes the transition. The FIN case keeps making progress, one pass later and still without noisy logging. Any other state keeps today's rds_conn_path_error() handling; no current cp_state writer can leave a DISCONNECTING path in anything but RDS_CONN_ERROR (every other writer is a cmpxchg from a non-DISCONNECTING state), so that branch is defensive. On kernels without the preceding patches the same hazard exists

CVE	Severity	CVSS	CVSS Vector	Component	Description
					with the sample-based quiesce; the fix applies there equally.
CVE-2026-98072	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net/rds: use wq_has_sleeper() in release_in_xmit() release_in_xmit() clears RDS_IN_XMIT with clear_bit_unlock() and then checks waitqueue_active() to decide whether anyone needs waking. clear_bit_unlock() is only a release operation: it orders the critical section before the bit clear, but does not order the subsequent plain load of the wait queue head after it. The waiter side does the mirror image - it adds itself to the wait queue and then tests the bit. That is the classic store-buffering pattern: the releasing CPU can read the wait queue as empty while the waiting CPU still reads the bit as set, so the sleeper is never woken. The waiters are rds_conn_shutdown() and rds_tcp_reset_callbacks(), both in uninterruptible wait_event() with no timeout. A lost wake-up strands the shutdown worker on its single-threaded workqueue until some other sender releases the bit again - and on a connection that is being torn down precisely because it failed, there may never be another sender. The barrier used to be there: release_in_xmit() did clear_bit() followed by smp_mb__after_atomic() until commit 1422f28826d2 ("rds: introduce acquire/release ordering in acquire/release_in_xmit()") folded both into clear_bit_unlock(), which strengthened the lock hand-off but silently dropped the full barrier the wake-up check depends on. The refill counterpart, release_refill() in net/rds/ib_recv.c, still carries its smp_mb__after_atomic() for exactly this reason. Use wq_has_sleeper(), which is waitqueue_active() preceded by the required full barrier.
CVE-2026-98074	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bonding: do not clear curr_active_slave prematurely when releasing all slaves When releasing all slaves during bond destruction (all == true), __bond_release_one() unconditionally clears bond->curr_active_slave to NULL in every iteration. If a backup slave is released before the active slave, bond_alb_deinit_slave() triggers rlb_teach_disabled_mac_on_primary(), which increments the active slave dev promiscuity counter and sets bond_info->primary_is_promisc = 1. Because bond->curr_active_slave was prematurely cleared to NULL when releasing the backup slave, the subsequent iteration releasing the active slave evaluates oldcurrent as NULL, so bond_change_active_slave(bond, NULL) is skipped. Consequently, bond_alb_handle_active_change() is never called to decrement the promiscuity counter, permanently leaking promiscuous mode on the physical device after bond teardown. When oldcurrent == slave, bond_change_active_slave(bond, NULL) already sets bond->curr_active_slave to NULL. We only need to avoid selecting a new active slave when all == true. Replace the if (all) branch with if (!all && oldcurrent == slave).

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98075	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: reject BPF_PSEUDO_FUNC reference to the main program fixups.c:jit_subprogs() rewrites BPF_PSEUDO_FUNC loads to contain real function addresses. This function is invoked from bpf_jit_subprogs() only when env->subprog_cnt > 1. Meaning that for any program like below: int main(void *ctx) { void *ptr = main; ... bpf_timer_set_callback(..., ptr); ... } The 'ptr' won't be ever converted to contain an address. In combination with e.g. bpf_timer_set_callback() this would lead to a function call at a bogus address. Instead of complicating the implementation, just assume that no useful program needs main to be a sync or async callback and reject BPF_PSEUDO_FUNC loads for the main subprogram.
CVE-2026-98078	MEDIUM	4.4	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:L	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ipvs: fix reversed sequence option serialization hton_seq() expects the host-order source first and the unaligned network-order destination second. The version 1 sync sender passes these arguments in reverse for both sequence blocks. This leaves 24 bytes of the kmalloc-backed message unwritten. It may disclose stale heap data and replace the live connection sequence state with values read from the buffer. Pass the connection sequence state as the source and the message payload as the destination for both blocks.
CVE-2026-98079	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: btrfs: zstd: fix lost wakeup when waiting for a workspace A writer can sleep forever in zstd_get_workspace() even though a workspace is free. When zstd_alloc_workspace() fails, the task is queued on zwsm->wait and schedules unconditionally, never re-testing the pool. zstd_put_workspace() publishes the workspace and then calls cond_wake_up(), which only wakes when a sleeper is already visible, so a workspace returned between the failed allocation and prepare_to_wait() wakes nobody. The window is wide: zstd_alloc_workspace() goes through kvmalloc() and may enter reclaim. Only a max level workspace triggers the wakeup and one is deliberately kept allocated as the fallback every waiter waits for, so once its wakeup is lost the writer stays in TASK_UNINTERRUPTIBLE until some other task happens to return one. Re-check the pool after prepare_to_wait() has published the waiter, and use the workspace if one turned up.
CVE-2026-98080	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: btrfs: do not force reloc root creation during qgroup_account_snapshot() [BUG] When running btrfs/252 with quota enabled through MKFS_OPTIONS="-O quota", it has a high chance to trigger the following kernel warning and flips the fs RO: BTRFS info (device dm-2): relocating block group 30408704 flags metadata|dup ------------[cut here]------------ WARNING: fs/btrfs/extent-tree.c:879 at lookup_inline_extent_backref+0x74b/0x960 [btrfs], CPU#4: btrfs/2173 CPU: 4 UID: 0 PID: 2173 Comm: btrfs Not tainted 7.2.0-rc6-custom+ #457 PREEMPT(full) 3adc6528fb66f7a55fe1095385818e742f200aab Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS unknown 02/02/2022 RIP: 0010:lookup_inline_extent_backref+0x74b/0x960 [btrfs] Call Trace: insert_inline_extent_backref+0x7c/0x160 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] __btrfs_inc_extent_ref+0xa9/0x270 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] __btrfs_run_delayed_refs+0x4af/0x11c0 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] btrfs_run_delayed_refs+0x9d/0xf0 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72]

CVE	Severity	CVSS	CVSS Vector	Component	Description
					create_pending_snapshot+0x39d/0xf00 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] create_pending_snapshots+0x9b/0xc0 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] btrfs_commit_transaction+0x280/0xeb0 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] prepare_to_relocate+0x147/0x200 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] relocate_block_group+0x6b/0x5e0 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] btrfs_relocate_block_group+0x92c/0x2380 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] btrfs_relocate_chunk+0x3f/0x1a0 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] btrfs_balance+0xa2c/0x19c0 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] btrfs_ioctl+0x2839/0x2d30 [btrfs 32f09462c54d9c922fca74a3e4866f4aa7737b72] __x64_sys_ioctl+0x416/0x9a0 do_syscall_64+0xe1/0x790 entry_SYSCALL_64_after_hwframe+0x4b/0x53 ---[end trace 0000000000000000]--- BTRFS info (device dm-2): leaf 4593991680 gen 233 total ptrs 175 free space 5953 owner 2 BTRFS info (device dm-2): refs 3 lock_owner 2173 current 2173 item 0 key (166772736 METADATA_ITEM 1) itemoff 16250 itemsize 33 extent refs 1 gen 222 flags 2 ref#0: tree block backref root 266 [Skip the tree dump] item 174 key (263225344 METADATA_ITEM 0) itemoff 10328 itemsize 33 extent refs 1 gen 162 flags 258 ref#0: tree block backref root 267 BTRFS error (device dm-2): extent item not found for insert, bytenr 179847168 num_bytes 16384 parent 4594335744 root_objectid 273 owner 0 offset 0 BTRFS error (device dm-2): failed to run delayed ref for logical 179847168 num_bytes 16384 type 182 action 1 ref_mod 1: -117 [CAUSE] The above error is showing that there is a tree reference to a metadata extent that is no longer there. With "ref_verify" mount option (requires CONFIG_BTRFS_DEBUG), there is some extra debug output: BTRFS error (device dm-2): dumping block entry [180961280 16384], num_refs 0, metadata 1, from disk 0 BTRFS error (device dm-2): root entry 256, num_refs 18446744073709551615 BTRFS error (device dm-2): root entry 273, num_refs 18446744073709551615 BTRFS error (device dm-2): Ref action 3, root 273, ref_root 273, parent 0, owner 0, offset 0, num_refs 1 btrfs_force_cow_block+0x129/0x7d0 [btrfs] btrfs_cow_block+0x10a/0x250 [btrfs] btrfs_search_slot+0x5eb/0xf40 [btrfs] btrfs_insert_empty_items+0x3a/0x70 [btrfs] insert_with_overflow+0x53/0x130 [btrfs] btrfs_insert_dir_item+0x125/0x290 [btrfs] btrfs_add_link+0xaa/0x410 [btrfs] btrfs_rename+0x5ea/0xcd0 [btrfs] btrfs_rename2+0x28/0x60 [btrfs] vfs_rename+0x5b2/0xe10 filename_renameat2+0x244/0x430 __x64_sys_rename+0x48/0x70 do_syscall_64+0xe1/0x790 entry_SYSCALL_64_after_hwframe+0x4b/0x53 ---truncated---
CVE-2026-98081	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: btrfs: zoned: finish active block group cleanup if call_zone_finish() fails do_zone_finish() clears BLOCK_GROUP_FLAG_ZONE_IS_ACTIVE before finishing the zones. If call_zone_finish() then fails it returned early, leaving the now inactive block group on fs_info->zone_active_bgs, leaking its reference, the BTRFS_FS_NEED_ZONE_FINISH waiters are never woken, and as its alloc_offset equals the zone capacity btrfs_zone_finish_one_bg() keeps selecting it, spinning btrfs_zoned_activate_one_bg(). Fall through to the cleanup on failure too and return the error, but keep the block group read-only as its zones are left inconsistent.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98082	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: btrfs: fix the possible bioc_list memory leak during error There are two possible ways to leak bioc memory on btrfs_ordered_extent::bioc_list: - An error occurred for btrfs_insert_one_raid_extent() Then the function btrfs_insert_raid_extent() immediately return without freeing any bioc in the bioc_list. - An ordered extent hit an IO error In that case the ordered extent will have BTRFS_ORDERED_IOERR set, and skip the call on btrfs_insert_raid_extent() completely. Fix the problem by: - Introduce a new helper, btrfs_cleanup_ordered_bioc_list() Which will remove all bioc from the bioc_list, and release the bioc. - Call the above helper for btrfs_insert_raid_extent() So that the cleanup helper is always called no matter what. - Call the above helper for btrfs_finish_one_ordered() This is called just before the final release on the ordered extent. This was reported by Sashiko when reviewing another patch.
CVE-2026-98084	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: backtracking shouldn't clear outer frame R1-R5 for callbacks When processing calls to bpf_loop() verifier marks R1 (and R4) as precise. R1 tracks loop iterations number and because of the 'callback_depth < R1' mechanics in check_helper_call() must be marked precise. However, precision propagation for R1 was broken, when bpf_loop() call was verified on a second iteration. Consider the following verification trace: - main: bpf_loop(nr_loops, callback ...) - callback: BPF_EXIT - main: bpf_loop(nr_loops, callback ...) - ... While the first visit of the call to bpf_loop() propagated R1 precision as expected, the second call to mark_chain_precision() in the check_helper_call() set R1, but it was immediately reset when backtrack_insn() processed preceding BPF_EXIT in the loop deleted in this patch. Because of that, the second visit of the call to bpf_loop() injected checkpoint with R1 not marked as precise. Which could trick the verifier into accepting unsafe programs. See the next patch for an example of such program. Commit is structured in a way to minimize conflicts when 'bpf' would be eventually merged with 'bpf-next'.
CVE-2026-98086	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ALSA: ump: do not touch legacy_rmidi before it exists snd_ump_parse_endpoint() sets ump->parsed on every exit, including error, before the caller attaches the legacy rawmidi device. ump_handle_ep_name_msg() then treats parsed as "legacy_rmidi is live" and calls ump_legacy_set_rawmidi_name(), which snprintf()s into ump->legacy_rmidi->name. If a UMP packet arrives in that window (IRQ path from snd_ump_receive), legacy_rmidi is still NULL (KASAN null-ptr-deref in snprintf). Guard the legacy helpers. parsed only means endpoint info was parsed, not that legacy_rmidi exists.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98087	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: sched/rt,dl: Skip migrate-disabled tasks when picking a push candidate A migrate_disable()'d RT task cannot be moved to another CPU, but the scheduler still keeps such a task on that CPU's pushable list (rq->rt.pushable_tasks) and still marks the runqueue RT-overloaded (rq->rt.overloaded = 1). So the RT balancer keeps treating this CPU as having a task to move away, and keeps trying to move the task, but the push can never succeed. When the head is pinned, push_rt_task() does not give up either. It falls back to pushing rq->curr instead, using the per-CPU stopper, as added by commit a7c81556ec4d ("sched: Fix migrate_disable() vs rt/dl balancing"). The CPU spends tens of milliseconds in this retry loop. The core is isolated for real-time work, but during the loop nearly half of its time is consumed by pushes that cannot succeed. An ftrace capture of the affected CPU, with sched_switch enabled and commit 94894c9c477e ("sched/rt: Skip currently executing CPU in rto_next_cpu()") applied, shows where the CPU time went. Two SCHED_FIFO tasks at equal priority shared the CPU, taskA migrate_disable()'d and queued, taskB as rq->curr. In one 89 ms window, taskB got only 52 ms of CPU. The other 37 ms went to the stopper thread. The scheduler kept trying to push taskA, the pinned head of the pushable list, fell back to pushing taskB instead, and woke the stopper 5204 times. Every one of those pushes failed and no task was moved. taskA stayed runnable and queued the whole time, and never ran. Pushing taskB fails on a re-check. find_lock_lowest_rq() drops the rq lock to take the target rq lock, then checks again with "task != pick_next_pushable_task(rq)". The task being pushed is taskB, but the pick returns taskA, the head of the pushable list. taskB is rq->curr, and set_next_task_rt() removes the running task from that list, so taskB can never be the head. The check expects a candidate taken from the pushable list, but the fallback pushes rq->curr, which is never on that list. So the check fails every time. .--> push-IPI arrives | | v | pushable head = taskA -> pinned, cannot be pushed | | v | so push taskB instead -> wake migration/N, a stop-class | | thread, so it preempts taskB | v | re-check compares taskB against the pushable head, | which is still taskA -> give up | | v | nothing moved, taskA still queued, rq still overloaded | | '---------' repeats every ~17 us, 5204 times, for 89 ms The loop cannot stop itself. Every round leaves the runqueue exactly as it was, so the next push-IPI does the same thing. In the capture it ended only when taskB went to sleep on its own. taskA was then picked locally and left the pushable list. CPU time per task in the window, from sched_switch: taskB 51.95 ms real work migration/N 37.18 ms nothing moved taskA 0.00 ms queued the whole time, never picked idle 0.01 ms Counts over the same window: 7667 push-IPIs handled on this CPU 17481 pick_next_pushable_task() returned taskA, still pinned 5204 find_lock_lowest_rq() gave up on the re-check 1 push that actually completed 0 migrations of taskA The CPU times and the window length come from the standard sched_switch tracepoint. The counts needed tracepoints added inside the RT balancer for this investigation. The self-IPI path is closed by the rto_next_cpu() fix above, and that part works. But the runqueue is still marked overloaded, because the pinned task is still advertised as pushable. Other CPUs now send the push-IPIs during their own RT balancing, and the same loop runs again. Closing the self-IPI path did not stop a pinn ---truncated---

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98089	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bonding: alb: fix uninitialized transport header access in alb_determine_nd() alb_determine_nd() uses icmp6_hdr(skb) to inspect ICMPv6 headers. However, in xmit paths (e.g. packets sent via AF_PACKET / raw sockets or forwarded packets), skb->transport_header is not guaranteed to be initialized. While pskb_network_may_pull() ensures the packet data is linear starting from the network header, it does not set or adjust the transport header offset. Dereferencing icmp6_hdr(skb) can therefore access out-of-bounds memory. Fetch the icmp6hdr directly after ipv6hdr following pskb_network_may_pull(), and reload ipv6hdr in case pskb_may_pull() reallocated skb->head. Also remove the unused bond argument from alb_determine_nd().
CVE-2026-98090	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: btrfs: restore active device pointers after failed sprout btrfs_init_new_device() switches latest_dev and possibly s_bdev from the seed device to the new sprout device before creating the first writable chunks. If chunk creation or the subsequent sprout setup fails, the error path releases the new device without switching those pointers back. btrfs_show_devname() can then dereference the freed latest_dev and crash. Restore the active device pointers to the latest seed device before removing and releasing the failed sprout device.
CVE-2026-98091	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: btrfs: detach failed sprout device from transaction update list When creating the first metadata chunk for a sprout filesystem, create_chunk() adds the new device to the transaction dev_update_list through device->post_commit_list. If the subsequent system chunk creation fails, btrfs_init_new_device() aborts the transaction and releases the device while post_commit_list is still linked. This triggers a warning in btrfs_free_device() and leaves the transaction list referencing freed memory. Detach the device while holding chunk_mutex before releasing it.
CVE-2026-98092	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ASoC: amd: yc: fix memory leak in acp6x_pdm_dma_close() acp6x_pdm_dma_close() does not free the runtime->private_data buffer allocated in acp6x_pdm_dma_open(). Add the missing kfree.
CVE-2026-98094	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: staging: fbtft: make dirty_lock IRQ-safe fbtft_mkdirty() can be reached from the fbcon rendering path while processing printk() in hardirq context. Meanwhile, dirty_lock is also taken by fbtft_deferred_io() in workqueue context with local interrupts enabled. Lockdep reports a possible IRQ lock inversion involving dirty_lock and console_owner. A hardirq can interrupt a CPU holding dirty_lock and enter the console rendering path, which can attempt to acquire dirty_lock again. The following lockdep report was observed on an RK3566 system with CONFIG_PROVE_LOCKING enabled: WARNING: possible irq lock inversion dependency detected swapper/2/0 just changed the state of lock: (console_owner){-...}-{0:0} but this lock took another, HARDIRQ-unsafe lock in the past: (&par->dirty_lock){+.+.}-{2:2} CPU0 CPU1 ---- ---- lock(&par->dirty_lock); local_irq_disable(); lock(console_owner); lock(&par->dirty_lock); lock(console_owner); *** DEADLOCK *** Use spin_lock_irqsave() for fbtft_mkdirty() and spin_lock_irq() for fbtft_deferred_io(). They only access the dirty line range, so the IRQ-off regions remain short.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98095	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: af_packet: Don't cast tpacket_hdr.tp_len to int in tpacket_parse_header(). syzbot reported BUG() in sock_sendmsg_nosec(). [0] The problem is that tpacket_parse_header() casts user-provided tpacket_hdr.tp_len, which is u32, to int. If the length is larger than INT_MAX, the following condition in tpacket_parse_header() passes, if (unlikely(tp_len > size_max)) and any negative value can be returned to the caller, up to sock_sendmsg_nosec(). The repro set tpacket_hdr.tp_len to 0xffffffef, which is cast to -EIOCBQUEUED (-529), triggering BUG() in sock_sendmsg_nosec(). * (uint64_t*)0x200000000008 = 0xffffffef; ... syscall(__NR_write, /*fd=*/r[0], /*buf=*/0x200000000000ul, /*count=*/1ul); Let's define the local tp_len as u32 in tpacket_parse_header(). [0]: kernel BUG at net/socket.c:803! Oops: invalid opcode: 0000 [#1] SMP KASAN PTI CPU: 0 UID: 0 PID: 5628 Comm: syz-executor176 Not tainted syzkaller #0 PREEMPT(full) Hardware name: Google Google Compute Engine/Google Compute Engine, BIOS Google 07/24/2026 RIP: 0010:sock_sendmsg_nosec+0x145/0x180 net/socket.c:803 Code: 06 67 48 0f b9 3a eb 95 e8 e8 3a 22 f8 48 89 df 4c 89 f6 4c 89 e2 4d 89 fb 2e e8 32 a5 5c 16 e9 51 ff ff ff e8 cc 3a 22 f8 90 <0f> 0b e8 c4 3a 22 f8 48 83 c3 18 48 89 d8 48 c1 e8 03 42 80 3c 28 RSP: 0018:ffffc90003aefb48 EFLAGS: 00010293 RAX: ffffffff89a578d4 RBX: ffff8880764c67c0 RCX: ffff88807fb23e80 RDX: 0000000000000000 RSI: 00000000ffffffef RDI: 00000000ffffffef RBP: 00000000ffffffef R08: ffffc90003aef747 R09: 1ffff9200075dee8 R10: dffffc0000000000 R11: fffff5200075dee9 R12: 0000000000000001 R13: dffffc0000000000 R14: ffffc90003aefbc0 R15: ffffffff8aac4310 FS: 000055559101b400(0000) GS:ffff888124ce0000(0000) knlGS:0000000000000000 CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033 CR2: 0000200000000210 CR3: 0000000073dca000 CR4: 00000000003526f0 Call Trace: __sock_sendmsg net/socket.c:815 [inline] sock_write_iter+0x2de/0x3e0 net/socket.c:1266 new_sync_write fs/read_write.c:595 [inline] vfs_write+0x612/0xba0 fs/read_write.c:687 ksys_write+0x150/0x270 fs/read_write.c:739 do_syscall_x64 arch/x86/entry/syscall_64.c:61 [inline] do_syscall_64+0x166/0x520 arch/x86/entry/syscall_64.c:84 entry_SYSCALL_64_after_hwframe+0x77/0x7f RIP: 0033:0x7f173130ecb9 Code: c0 79 93 eb d5 48 8d 7c 1d 00 eb 99 0f 1f 44 00 00 48 89 f8 48 89 f7 48 89 d6 48 89 ca 4d 89 c2 4d 89 c8 4c 8b 4c 24 08 0f 05 <48> 3d 01 f0 ff ff 73 01 c3 48 c7 c1 d8 ff ff ff f7 d8 64 89 01 48 RSP: 002b:00007ffd67e44248 EFLAGS: 00000246 ORIG_RAX: 0000000000000001 RAX: ffffffffffffffda RBX: 0000200000000000 RCX: 00007f173130ecb9 RDX: 0000000000000001 RSI: 0000200000000000 RDI: 0000000000000003 RBP: 0000000000000001 R08: 0000000000000000 R09: 0000000000000000 R10: 0000000000000000 R11: 0000000000000246 R12: 00007ffd67e44388 R13: 0000000000000002 R14: 00002000000000c0 R15: 0000000000000002
CVE-2026-98097	MEDIUM	5.3	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: tipc: Dont send random pad bytes in RESET/ACTIVATE messages The interface name is passed in a fixed length (TIPC_MAX_IF_NAME) buffer. Replace the strcpy(data, l->if_name) with memcpy() so that the pad bytes are actually written (l->if_name[] is zero padded) rather than sending random bytes from the skb to the remote system. Replace two other strcpy() with strscpy().

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98098	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: tipc: fix NULL deref in tipc_named_node_up() on empty publication list User-space applications can bind a large number of service addresses to one or more sockets. Each binding of a local-scope service address inserts one entry (publication) into the TIPC name table. If the number of these publications exceeds TIPC_MAX_PUBL (65535), protocol service types (such as node state and link state) are no longer inserted into the name table. This causes two issues: 1. User-space applications subscribing to node or link up/down events stop receiving notifications. 2. A NULL pointer dereference can occur: BUG: kernel NULL pointer dereference, address: 00000000000000d0 ... CPU: 0 UID: 0 PID: 0 Comm: swapper/0 Not tainted 7.2.0-rc4-default+ #5 PREEMPT(full) ... RIP: 0010:tipc_named_node_up (./include/linux/skbuff.h:2251 net/tipc/name_distr.c:195 net/tipc/name_distr.c:221) ... Call Trace: tipc_node_write_unlock (net/tipc/node.c:428) tipc_rcv (net/tipc/node.c:934 net/tipc/node.c:2189) tipc_udp_recv (net/tipc/udp_media.c:389) Thread 1 (tipc_net_finalize) | Thread 2 (named_distribute) ----------|---------- | ... | list_for_each_entry(publ, pls, binding_node) { | ... | __skb_queue_tail(list, skb); | ... | } | ... | hdr = buf_msg(skb_peek_tail(list)); ... | tipc_nametbl_publish(); | If 'tipc_nametbl_publish()' (Thread 1) fails because the number of local publications reaches TIPC_MAX_PUBL, list (Thread 2) will be empty. As a result, NULL is passed to 'buf_msg()', leading to a NULL pointer dereference. Fix these issues by allowing protocol service types (node state, link state, and topology server) to be inserted into the name table unconditionally. This ensures that users subscribing to these types always receive notifications. In addition, the maximum number of local user publications is reduced to (TIPC_MAX_PUBL - 1). This ensures that the maximum bulk size calculated in tipc_link_set_queue_limits() remains valid.
CVE-2026-98099	MEDIUM	4.7	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ipv6: mcast: use rcu_assign_pointer() for __rcu list updates Several places in net/ipv6/mcast.c update RCU-protected lists (np->ipv6_mc_list, idev->mc_list, idev->mc_tomb) using direct pointer assignments instead of rcu_assign_pointer(): 1. In __ipv6_dev_mc_dec(), unlinking a group from idev->mc_list did: *map = ma->next; without rcu_assign_pointer() while concurrent readers traverse idev->mc_list locklessly under rcu_read_lock(). 2. In ipv6_sock_mc_drop() and __ipv6_sock_mc_close(), unlinking a group from np->ipv6_mc_list directly assigned *lnk = mc_lst->next and np->ipv6_mc_list = mc_lst->next without rcu_assign_pointer(), racing with lockless readers in inet6_mc_check(). 3. In __ipv6_sock_mc_join(), mc_lst->next was initialized to np->ipv6_mc_list via raw assignment before publishing mc_lst. 4. In mld_del_delrec() and __ipv6_dev_mc_inc(), __rcu source pointers passed into rcu_assign_pointer() lacked explicit dereference helpers. Fix these by consistently using rcu_assign_pointer() along with mc_dereference() / sock_dereference().

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98101	MEDIUM	4.7	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ipv6: mcast: use copy-on-write RCU updates in ip6_mc_source() pmc->sflist is read locklessly under rcu_read_lock() by inet6_mc_check() during packet reception in the UDP and RAW multicast receive paths. ip6_mc_source() mutated psl->sl_addr and psl->sl_count in-place when adding or removing a source filter. Additionally, when expanding the filter buffer, newpsl was published via rcu_assign_pointer() before writing the new source into the array. Because 16-byte struct in6_addr writes are not atomic and array shifting is not synchronized with RCU readers, concurrent readers in inet6_mc_check() could read torn IPv6 addresses or observe duplicated/missed source entries. Fix this by switching ip6_mc_source() to copy-on-write RCU updates: allocate and fully populate newpsl before publishing it via rcu_assign_pointer(), and reclaim the old filter via kfree_rcu(), matching ip6_mc_msfilter(). Also remove the now unused IP6_SFBLOCK macro.
CVE-2026-98102	MEDIUM	4.7	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ipv6: mcast: fix RCU list diversion in ip6_mc_del1_src() When removing a source filter whose count reaches zero, ip6_mc_del1_src() unlinks psf from pmc->mca_sources. If the filter was previously active, the code moved psf directly into pmc->mca_tomb by updating psf->sf_next. Because pmc->mca_sources is traversed locklessly under RCU (e.g. by ipv6_chk_mcast_addr()), mutating psf->sf_next before a grace period elapses diverts concurrent readers to the tombstone list. Consequently, readers miss remaining active sources in pmc->mca_sources and improperly examine deleted tombstone entries. Fix this by allocating a new tombstone node for pmc->mca_tomb (as done in sf_setstate()) and retiring the original psf via kfree_rcu().

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98104	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net/sched: cls_u32: fix duplicate handle when node ID pool is exhausted gen_new_kid() falls back to returning max(htid | 0xFFF) when both idr_alloc_u32() ranges are full, instead of reporting an error. u32_change() trusts that value and inserts a new knode with a handle that is already live in the hash table, breaking handle uniqueness within the table's node ID space. The handle was never reserved in ht->handle_idr, so every later error path that does idr_remove(&ht->handle_idr, handle) removes the reservation of a different, live knode, which is then reused — one failed add compounds into further duplicates. The 4095 limit is per (table, bucket) — ht->handle_idr is per hash table and the range is derived from htid (bucketid), so a table with divisor 256 can legitimately hold 256*4095 knodes. The sibling helper gen_new_htid() has the same silent in-band failure: it returns 0 when the tp_c handle pool (1..0x7FF) is full, and u32_init() publishes the root hash table with handle 0 without checking. Two root tables with handle 0 alias in u32_lookup_ht(), allowing cross-tcf_proto knode add/lookup/delete. Add the same exhaustion check that the divisor path already has. Return an error so u32_change() fails with ENOSPC/ENOMEM when the node ID space is exhausted, and so u32_init() fails with -ENOMEM when the hash table ID space is exhausted. The extack message distinguishes pool exhaustion (-ENOSPC) from a transient allocation failure (-ENOMEM). Conditions to recreate the bug: - CONFIG_NET_SCHED=y, CONFIG_CLS_U32=y (or =m with module loaded) - Create a clsact qdisc on a device, then add 4095 u32 filters with auto-generated handles to fill the node ID space for the root hash table (single bucket). The 4096th auto-handle filter add triggers the duplicate handle (fh 800::fff reused). Reachable at Level 2 (unshare -Urn, namespace-local CAP_NET_ADMIN). - For gen_new_htid: create 2047 u32 proto entries on the same block to fill the tp_c handle pool, then create one more. The root table gets handle 0 and aliases with other handle-0 root tables.
CVE-2026-98105	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: ethernet: oa_tc6: Improve the error recovery When oversubscribed traffic causes lot of buffer overflow errors, probably due to loss of data chunks, driver fails to find a data chunk with end_valid bit set, before it runs out of sk buffer space. As a result, assert is seen during skb_put. Now, check is made if skb buffer has enough tailroom for the incoming data before accepting. If there is no room, current frame is abandoned and it will start looking for a data chunk with start_valid bit, that is a new frame. SK buffer allocation error is considered as recoverable error. rx_buf_overflow flag is too specific and no longer the only condition this flag is used for. Therefore it is renamed as wait_until_start_valid. This is more appropriate as this flag is used to look for the next data chunk with SV bit set, after failures like buffer overflow, buffer allocation failure, skb pointer validity besides buffer overflow error. Not writing to status0 if it reads 0.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98109	MEDIUM	4.7	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: Bluetooth: hci_core: Fix race condition during device registration In hci_register_dev(), the power_on work item is queued to hdev->req_workqueue before initializing hdev->adv_monitors_idr and registering the MSFT extension via msft_register(). For devices marked with quirks such as HCI_QUIRK_RAW_DEVICE, the HCI_UNCONFIGURED flag is set on the device. When the power_on work item runs concurrently on another CPU, hci_power_on() detects that the device is unconfigured and immediately invokes hci_dev_do_close(), which calls msft_do_close(). Concurrently, msft_register() allocates the msft structure and exposes it to hdev->msft_data prior to calling mutex_init(&msft->filter_lock). If msft_do_close() executes while hdev->msft_data is already assigned but the mutex has not yet been initialized, mutex_lock(&msft->filter_lock) operates on an uninitialized mutex, triggering a DEBUG_LOCKS warning: DEBUG_LOCKS_WARN_ON(lock->magic != lock) WARNING: kernel/locking/mutex.c:625 at __mutex_lock_common kernel/locking/mutex.c:625 [inline] WARNING: kernel/locking/mutex.c:625 at __mutex_lock+0x12d8/0x1550 kernel/locking/mutex.c:821 ... Call Trace: msft_do_close+0x308/0x7b0 net/bluetooth/msft.c:693 hci_dev_close_sync+0x86b/0x10a0 net/bluetooth/hci_sync.c:5522 hci_dev_do_close net/bluetooth/hci_core.c:499 [inline] hci_power_on+0x32c/0x750 net/bluetooth/hci_core.c:937 process_one_work kernel/workqueue.c:3322 [inline] process_scheduled_works+0xa8e/0x14e0 kernel/workqueue.c:3405 worker_thread+0x92d/0xe10 kernel/workqueue.c:3486 kthread+0x388/0x470 kernel/kthread.c:436 ret_from_fork+0x514/0xb70 arch/x86/kernel/process.c:158 ret_from_fork_asm+0x1a/0x30 arch/x86/entry/entry_64.S:245 Fix this by moving the queue_work() call in hci_register_dev() to after idr_init(&hdev->adv_monitors_idr) and msft_register(hdev) so that device structures and extensions are fully initialized before asynchronous tasks can access them. Additionally, assign hdev->msft_data in msft_register() only after mutex_init(&msft->filter_lock) has completed.
CVE-2026-98114	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ksmbd: propagate DACL parsing errors parse_dacl() silently accepts truncated ACEs and allocation failures, allowing set_info_sec() to continue with an incomplete ACL conversion. Return parsing and allocation errors to parse_sec_desc() so malformed security descriptors are rejected before inode attributes or ACL xattrs are updated.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98119	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfs: break unbuffered write when netfs_alloc_subrequest() fails syzbot reported a null-ptr-deref below [1] following a fault injection in netfs_alloc_subrequest(). [0] When netfs_alloc_subrequest() fails, subreq is NULL. Later, netfs_prepare_write() tries to initialize members of subreq(e.g., source), the issue in [1] is triggered. Let's handle the error of netfs_prepare_write() properly. [0] FAULT_INJECTION: forcing a failure. name failslab, interval 1, probability 0, space 0, times 0 Call Trace: netfs_alloc_subrequest+0x116/0x3f0 netfs_prepare_write+0x76/0x7b0 netfs_unbuffered_write+0x75c/0x2020 netfs_unbuffered_write_iter_locked+0x7d6/0xa80 netfs_unbuffered_write_iter+0x442/0x720 v9fs_file_write_iter+0xbf/0x100 vfs_write+0x6ac/0x1050 [1] KASAN: null-ptr-deref in range [0x00000000000000a8-0x00000000000000af] RIP: 0010:netfs_prepare_write+0xbc/0x7b0 fs/netfs/write_issue.c:173 Call Trace: netfs_unbuffered_write+0x75c/0x2020 fs/netfs/direct_write.c:111 netfs_unbuffered_write_iter_locked+0x7d6/0xa80 fs/netfs/direct_write.c:290 netfs_unbuffered_write_iter+0x442/0x720 fs/netfs/direct_write.c:382 v9fs_file_write_iter+0xbf/0x100 fs/9p/vfs_file.c:409 new_sync_write fs/read_write.c:595 [inline]
CVE-2026-98121	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: watchdog: msc313e: Fix NULL pointer dereference in PM callbacks msc313e_wdt_probe() doesn't set the driver data for the platform device. As a result, dev_get_drvdata() in msc313e_wdt_suspend() and msc313e_wdt_resume() will return NULL, leading to a NULL pointer dereference afterward. Set the platform device driver data in msc313e_wdt_probe().

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98123	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: sctp: fix soft lockup from unpadded ASCONF-ACK parameter iteration sctp_verify_asconf() walks ASCONF-ACK parameters with sctp_walk_params(), which advances by SCTP_PAD4(length), while the consumer sctp_get_asconf_response() iterates the same parameters advancing by the raw length, without padding. A single odd-length parameter desynchronises the two walks and makes the consumer interpret attacker-controlled bytes at a misaligned offset. When those bytes yield a length of zero, the while loop over asconf_ack_len makes no progress, spinning forever in softirq context, and the watchdog reports a soft lockup. All reads stay within the received skb, so the lockup is a pure remote denial of service. A remote peer can trigger it with a crafted ASCONF-ACK on an ADD-IP enabled association with an outstanding ASCONF (RFC 5061 section 4.1.2 requires the chunk to be authenticated, but the predefined empty key id 0 allows the peer to compute the same association HMAC from publicly exchanged parameters, so the gate does not help). The SCTP_PARAM_ERR_CAUSE case of sctp_verify_asconf() also performs no length check, letting a parameter without a complete error header reach the consumer, which reads errhdr.cause past the end of the parameter, an out-of-bounds read. Reject SCTP_PARAM_ERR_CAUSE parameters shorter than sizeof(struct sctp_addip_param) + sizeof(struct sctp_errhdr) at the verifier, and advance the consumer iterator with the same padding rule as the verifier to keep the two walks in lockstep. The verifier change guarantees a complete error header in every ERR_CAUSE parameter the consumer can see, so the consumer's asconf_ack_len check is dropped and it returns err_param->cause directly. The consumer padding fix is still required because odd lengths remain valid for SCTP_PARAM_ERR_CAUSE per RFC 5061. The issue was found by ZeroHive, a vulnerability hunting agent at Tencent Yunding Lab.
CVE-2026-98126	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: smb/client: validate new EOF for zero range When FALLOC_FL_ZERO_RANGE is used without FALLOC_FL_KEEP_SIZE, smb3_zero_range() may extend EOF without checking RLIMIT_FSIZE, allowing the file to grow beyond the caller's file-size limit. Fix this by calling inode_newsize_ok() before sending the zero-range request when the operation would extend EOF. Reproducer, using a file on a CIFS mount: bash -c ' FILE=/mnt/cifs/repro trap "" SIGXFSZ ulimit -f 3072 truncate -s 2M "$FILE" fallocate --zero-range -o 0 -l 4M "$FILE" echo "fallocate rc=$?" stat -c "file size=%s" "$FILE" ' Before this change, the operation succeeds despite the 3 MiB limit: fallocate rc=0 file size=4194304 After this change, fallocate fails and leaves the file at 2 MiB.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98127	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: smb/client: validate new EOF for insert range smb3_insert_range() does not check if the new file size (i_size + len) is valid. This allows FALLOC_FL_INSERT_RANGE to bypass RLIMIT_FSIZE, exceed s_maxbytes, or produce a size outside the loff_t range. Use check_add_overflow() to calculate the new EOF. Validate it with inode_newsize_ok() before modifying the file. Reproducer, using a file on a CIFS mount: bash -c ' FILE=/mnt/cifs/repro trap '"" SIGXFSZ ulimit -f 3072 # RLIMIT_FSIZE = 3 MiB # A regular write is stopped at 3 MiB. dd if=/dev/zero of="$FILE" bs=1M count=4 status=none stat -c "size after write: %s" "$FILE" # Insert 2 MiB into a 2 MiB file. truncate -s 2M "$FILE" fallocate -i -o 0 -l 2M "$FILE" stat -c "size after insert: %s" "$FILE" ' Before this change, the regular write stops at the 3 MiB limit, but insert range grows the file to 4 MiB: dd: error writing '/mnt/cifs/repro': File too large size after write: 3145728 size after insert: 4194304 After this change, insert range also fails at the limit and leaves the 2 MiB file unchanged: dd: error writing '/mnt/cifs/repro': File too large size after write: 3145728 fallocate: fallocate failed: File too large size after insert: 2097152
CVE-2026-98128	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: scsi: mpi3mr: Fix target device refcount leak in mpi3mr_sas_port_add() mpi3mr_get_tgtdev_by_addr() increments the target device kref when it returns a device. If a subsequent error triggers a goto out_fail after the tgtdev reference is acquired, the reference is never released because the out_fail path does not call mpi3mr_tgtdev_put(). This prevents the target device structure from ever being freed. Add a tgtdev put in the out_fail path, guarded by a NULL check since tgtdev is only acquired for SAS_END_DEVICE types and the same cleanup path is shared by earlier error cases where tgtdev is still NULL.
CVE-2026-98129	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: scsi: mpi3mr: Fix NULL pointer dereference in mpi3mr_sas_port_add() sas_port_alloc_num() can return NULL on memory allocation failure. The return value is passed directly to sas_port_add() without a NULL check, which causes a NULL pointer dereference. Additionally, if sas_port_add() fails, the allocated port is not freed before jumping to out_fail, leaking the sas_port structure. Call sas_port_free() to properly release it.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98142	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/cirrus-qemu: Validate BAR0 size during probe The `cirrus-qemu` driver relies on `CIRRUS_VRAM_SIZE` (4 MB) to validate framebuffer sizes. However, during PCI probe, the driver mapped BAR0 without verifying that its size matches `CIRRUS_VRAM_SIZE`. If a PCI device with a BAR0 smaller than 4 MB is bound to the driver, the mapped VRAM will be smaller than expected. Because validation checks assume 4 MB VRAM, framebuffers larger than the mapped memory can be created. When the display plane is updated (e.g. during release), `cirrus_primary_plane_helper_atomic_update()` copies the framebuffer to VRAM using `drm_fb_memcpy()`. Writing past the end of the mapped I/O memory causes a supervisor write page fault: BUG: unable to handle page fault for address: ffffc9000389c000 ... RIP: 0010:memcpy_toio+0x7c/0xe0 arch/x86/lib/iomem.c:110 ... Call Trace: iosys_map_memcpy_to include/linux/iosys-map.h:285 [inline] drm_fb_memcpy+0x325/0x5d0 drivers/gpu/drm/drm_format_helper.c:442 cirrus_primary_plane_helper_atomic_update+0x98a/0xb00 drivers/gpu/drm/tiny/cirrus-qemu.c:358 drm_atomic_helper_commit_planes+0x626/0xea0 drivers/gpu/drm/drm_atomic_helper.c:3038 drm_atomic_helper_commit_tail+0x60/0x510 drivers/gpu/drm/drm_atomic_helper.c:1989 commit_tail+0x2b1/0x3c0 drivers/gpu/drm/drm_atomic_helper.c:2074 drm_atomic_helper_commit+0xa77/0xb10 drivers/gpu/drm/drm_atomic_helper.c:2312 Fix this by validating in `cirrus_pci_probe()` that the PCI BAR0 resource is not less than `CIRRUS_VRAM_SIZE`, returning `-ENODEV` if it is less.
CVE-2026-98147	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: printk: Don't WARN on kthread_run failure. Since __kthread_create_on_node() returns -EINTR upon SIGKILL, we should not use WARN_ON() in order to catch kthread_run() failure.
CVE-2026-98148	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/gud: validate GUD_ROTATION_0 is present in supported rotations The rotation argument to drm_plane_create_rotation_property() is set to DRM_MODE_ROTATE_0, and the device reported rotation bitmask is used as the supported_rotations argument. The driver never validates that GUD_ROTATION_0 is present, so a device that omits it from its GUD_PROPERTY_ROTATION triggers the WARN_ON(rotation & ~supported_rotations) in drm_plane_create_rotation_property() Fix this by skipping the creation of rotation property if the device doesn't have the GUD_ROTATION_0 bit

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98151	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: Fix REG INVARIANTS VIOLATION on speculative pointer arithmetic Take the following unprivileged program as an example: r0 = bpf_map_lookup_elem(...) /* PTR_TO_MAP_VALUE, offset 0 */ ... 14: r0 += r1 /* r1 is a bounded scalar */ 15: r9 = r0 Loading it triggers a verifier warning from reg_bounds_sanity_check(): verifier bug: REG INVARIANTS VIOLATION (alu): const subreg tnum out of sync with range bounds r64= {.base=0x0, .size=0x0} r32={.base=0x0, .size=0xffffffff} var_off=(0x0, 0x0) What happens: 1. Processing insn 14 (r0 += r1) in adjust_ptr_min_max_vals(), the new offset is computed into dst_reg's var_off and 32/64-bit ranges. 2. Because pointer registers do not track 32-bit subregister bounds, __mark_reg32_unbounded() first sets r32 to the full range; r32 is re-derived from the offset at the end of the function by reg_bounds_sync(). 3. On the unprivileged path, sanitize_ptr_alu() is called and, via sanitize_speculative_path() -> push_stack(), snapshots the current register state and schedules the next instruction (insn 15) to be verified directly as a speculative path. 4. That snapshot is taken between step 2 and the final reg_bounds_sync(): at this point dst_reg's var_off still holds the (const) original offset while r32 has just been blanked to the full range, i.e. the two are out of sync. When the speculative path later verifies insn 15 (r9 = r0), the inconsistent state reaches reg_bounds_sanity_check() and trips the warning. var_off and the 32-bit range must always be consistent. There are two ways to keep the snapshot consistent: 1. sync var_off and r32 before the snapshot so they match, or 2. leave r32 at its original (already consistent) value and blank it only after the snapshot. The whole point of sanitize_ptr_alu() is to insert a harmless masking sequence that keeps the access in bounds under speculation, so the state it snapshots should faithfully represent that. Take approach 2: move __mark_reg32_unbounded() to after sanitize_ptr_alu(), so the speculative snapshot keeps the pointer's original, consistent r32. The non-speculative path is unchanged: r32 is still blanked before the offset is applied and re-derived by reg_bounds_sync().
CVE-2026-98152	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: nvmet-rdma: fix queue leak when connect backlog is exceeded When pending disconnecting queues exceed the backlog limit, the connect path only drops the device reference and leaks the newly allocated queue and its IB resources.
CVE-2026-98157	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: EDAC/device_sysfs: Use kstrtouint() for poll_msec to prevent truncation The poll_msec sysfs store file uses simple_strtoul() which accepts an unsigned long, but the target field (poll_msec) is unsigned int. On 64-bit systems, a value > UINT_MAX is silently truncated when stored. Fix the mismatch by using kstrtouint() instead. This rejects values larger than UINT_MAX at parse time, making truncation impossible. Also add a check for value < 1 to reject the 0-delay case, which would cause the poll work to spin without delay and consume 100% CPU.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-98158	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: ppp_async: drop the errored frame instead of resetting its headroom ppp_receive_nonmp_frame() prepends a two-byte direction tag before running the pass/active BPF filters: *(__be16 *)skb_push(skb, 2) = htons(PPP_FILTER_INBOUND_TAG); Nothing on the receive path guarantees those two bytes of headroom. The frame-error path in ppp_async's process_input_packet() resets a reused skb's headroom to zero while claiming to restore it to a freshly allocated state - but a fresh skb from dev_alloc_skb() carries NET_SKB_PAD: err: if (skb) { /* make skb appear as freshly allocated */ skb_trim(skb, 0); skb_reserve(skb, - skb_headroom(skb)); } ap->rpkt still points at that skb, so the next frame is reassembled into it with no headroom at all. A peer that sends a bad-FCS frame followed by one beginning ff 03 then leaves a single byte of headroom by the time the filter tag is pushed, which lands one byte below skb->head: skbuff: skb_under_panic: len:49 put:2 head:ffff888003c10000 data:ffff888003c0ffff tail:0x30 end:0x640 dev: kernel BUG at net/core/skbuff.c:214! RIP: 0010:skb_panic+0x13e/0x230 Call Trace: skb_push+0xbd/0x100 ppp_receive_nonmp_frame+0x48a/0x1d10 ppp_input+0x4e9/0x2f80 ppp_async_process+0x2a/0xe0 tasklet_action_common+0x20f/0x8a0 handle_softirqs+0x18e/0x590 Kernel panic - not syncing: Fatal exception in interrupt Zeroing the headroom violates the NET_SKB_PAD guarantee that dev_alloc_skb() gives the rest of the receive path. Besides the filter panic above, when CCP compression is enabled ppp_decompress_frame() hands skb->data - 2 to ->decompress()/->incomp(), which then reads out of bounds before skb->head for the same reason. Rather than restore the headroom, drop the errored frame - as ppp_synctty already does on its error path - and clear ap->rpkt so the next frame is reassembled into a fresh skb with proper headroom. This is simpler and fixes both the filter under-panic and the CCP out-of-bounds read. The original V1 of this patch made room in ppp_receive_nonmp_frame() with skb_cow_head(); Eric pointed out that fixing the root cause in the transport is the right approach. Found by fuzzing the PPP receive path with a mutating peer on a pty; it is an interesting (remote) DoS: root configures PPP, the peer supplies two crashing frames. The reproducer (repro-ppp-skb.c, unchanged from v1) panics in about a second, and returns cleanly with this applied.
CVE-2026-98159	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: mt76: mt7921: validate CLC firmware records The CLC region is supplied by firmware, but the loader trusts the region count and each record length. A malformed image can make the region table pointer precede the firmware buffer, make the record loop fail to advance, or index phy->clc past its end. Validate the table and record bounds before dereferencing or copying.
CVE-2026-100070	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: netfilter: nf_nat_sip: rewind offset when NAT shrinks the packet sashiko says: If map_addr() changes the packet length, such as when the public NAT IP string is shorter or longer than the internal IP, coff will still point to the offset relative to the pre-mangled packet. If the packet shrinks, coff could overshoot the correct position, potentially causing the next ct_sip_parse_header_uri() call to silently skip bytes and miss subsequent Contact headers. Could this lead to a failure to NAT those subsequent headers and leak internal network details?

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-100071	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: net: hsr: free learned nodes on device setup failure hsr_dev_finalize() can fail after a lower-device RX handler has already been registered (slave A is added before the failable slave B and interlink adds). RX handlers run in softirq regardless of the master's state, so frames received in that window can learn dynamic nodes into node_db, and the error unwind never releases them. Free both owned dynamic databases in the unwind, mirroring hsr_dellink(). proxy_node_db is provably empty on every current error exit (only interlink RX feeds it, and the interlink add is the last failable step) and is freed for symmetry. The order is safe: hsr_del_port() unregisters each RX handler with synchronize_net() before hsr_del_nodes() runs, which removes remaining entries with list_del_rcu() and defers their release with call_rcu() for readers already under RCU.
CVE-2026-100074	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: bpf: Mark bpf_refcount field as unique BPF_REFCOUNT is not marked as a unique field, while it should be. Fix this oversight.
CVE-2026-100076	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: staging: rtl8723bs: fix xmit_frame/xmit_buf leaks on mgnt-frame error paths issue_beacon(), issue_probersp() and issue_asocrsp() obtain a management xmit_frame together with its xmit_buf from the driver's fixed-size management-TX pools via alloc_mgtxmitframe(). On the normal path the frame is handed to dump_mgntframe(), which transfers ownership and eventually returns both objects to their pools (the frame and, for beacons, the buf in rtl8723bs_mgnt_xmit(); other bufs via the pending-xmitbuf/TX-completion path). Several error/edge paths return early after a successful alloc_mgtxmitframe() but before dump_mgntframe(), so ownership is never transferred and neither object is freed: - issue_beacon(): beacon larger than 512 bytes - issue_probersp(): cur_network->ie_length > MAX_IE_SZ - issue_probersp(): kzalloc() of the SSID scratch buffer fails - issue_asocrsp(): pkt_type is neither ASSOCRSP nor REASSOCRSP Because alloc_mgtxmitframe() removes the frame and buf from their free lists (list_del_init) without placing them on any pending list, an orphaned pair is on no list and referenced by nobody, so it is only reclaimed at driver teardown. Repeated hits progressively exhaust the management-TX pools until alloc_mgtxmitframe() returns NULL and the interface can no longer send beacons or probe/assoc responses. Free the frame and buffer on these paths, matching the existing correct error handling in issue_assocreq().
CVE-2026-100077	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: drm/msm: Recover HW before retire hung submit During recovery, it is not safe to retire the hung submit before we recover the GPU. Retiring the submit triggers BO free and that can result in GPU pagefaults since the GPU may be actively accessing those BOs. To fix this, retire the submits after gpu recovery is complete in recover_worker(). Patchwork: https://patchwork.freedesktop.org/patch/730655/
CVE-2026-100078	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: wifi: iwlwifi: mei: pass correct argument to function The first argument to iwl_mei_write_cyclic_buf() should be the cldev but the q_head pointer is passed instead. Fix it.

CVE	Severity	CVSS	CVSS Vector	Component	Description
CVE-2026-100079	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: usb: typec: ucsi: unregister debugfs entries on teardown ucsi_register() creates per-instance debugfs entries, but ucsi_unregister() keeps them around until ucsi_destroy(). Drivers like ucsi_glink that unregister/register the same UCSI instance across remoteproc restart then try to create an already existing debugfs directory and log: debugfs: 'pmic_glink.ucsi.0' already exists in 'ucsi' Unregister debugfs entries as part of ucsi_unregister(), and clear ucsi->debugfs after freeing it so repeated unregister paths remain safe.
CVE-2026-98160	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: staging: rtl8723bs: fix mismatched free of HalData in rtw_sdio_if1_init() padapter->HalData is allocated via vzalloc(), but incorrectly freed using kfree() in the rtw_sdio_if1_init() error path. Using kfree() to release this vmalloc-backed buffer can lead to memory corruption. Use rtw_hal_data_deinit() to pair the free correctly and free HalData with vfree(). The bug was first flagged by an experimental static analysis tool we are developing for kernel memory-management bugs. Manual inspection confirms that the issue is still present in current mainline. An x86_64 allyesconfig build showed no new warnings. As we do not have suitable RTL8723BS SDIO hardware to test with, no runtime testing was able to be performed.
CVE-2026-98161	MEDIUM	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	linux-rittal 6.12.100+git	In the Linux kernel, the following vulnerability has been resolved: nvdimm: pmem: keep PREFLUSH before data writes pmem_submit_bio() records a REQ_PREFLUSH error, but continues to copy the bio data and can later overwrite the error with a successful REQ_FUA flush. That lets data writes run after a failed preflush and can complete the bio successfully despite the failed ordering barrier. Run the REQ_PREFLUSH flush synchronously before touching the bio data and complete the bio with the flush error if it fails. Keep asynchronous flush chaining for REQ_FUA. At that point, data copy has completed and the parent bio can wait for the chained flush bio.
CVE-2025-66382	LOW	2.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L	expat 2.6.4	In libexpat through 2.7.3, a crafted file with an approximate size of 2 MiB can lead to dozens of seconds of processing time.
CVE-2026-41080	LOW	2.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L	expat 2.6.4	libexpat before 2.8.0 uses insufficient entropy, and thus hash flooding can occur via a crafted XML document.
CVE-2026-45186	LOW	2.9	CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L	expat 2.6.4	In libexpat before 2.8.1, the computational complexity of attribute name collision checks allows a denial of service via moderately sized crafted XML input.
CVE-2026-11979	LOW	7.8	CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H	libxml2 2.12.10	libxml2 is vulnerable to multiple stack-based buffer overflows in the xmlcatalog utility when running in --shell mode. The usershell() function processes user input using fixed-size stack buffers without proper bounds checking. By supplying an overly long input line, an attacker can overflow internal buffers (command, arg, and argv) during input parsing. This results in memory corruption within the stack frame. Successful exploitation may cause a crash or potentially allow arbitrary code execution in the context of the xmlcatalog process. This issue has been fixed in the commit c2e233fc. NOTE: The maintainers of this project did not agree that this issue is a vulnerability and considered it a bug.
CVE-2026-60000	LOW	3.7	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L	openssh 9.6p1	sshd in OpenSSH before 10.4 allows remote attackers to cause a denial of service (resource consumption from excessive authentication attempts) because MaxAuthTries was mishandled for GSSAPIAuthentication.

Recommended Actions

Rittal recommends updating affected devices to firmware version **v10.3.0** or later to address the vulnerabilities listed in this bulletin.

Contact

For questions regarding this bulletin, please contact:
Rittal PSIRT – psirt@rittal.com

Disclaimer

This security bulletin is provided "as is" without warranty of any kind. Rittal does not accept any liability for damages resulting from the use of this information. The information in this bulletin is subject to change without notice. Users are advised to verify the applicability of this information to their specific environment. This document may be updated as new information becomes available.