MLEE: Effective Detection of Memory Leaks on Early-Exit Paths in OS Kernels

Wenwen Wang

UNIVERSITY OF GEORGIA

Memory leaks can cause serious system issues

Performance issues

➔ Increased system response time due to reduced memory resources

Reliability issues

➔ Unavoidable system reboots when leaked objects exhaust memory resources

Security issues

➔ Exploited to launch security attacks, e.g., CVE-2019-12379 and CVE-2019-8980

Many memory leak detectors have been developed

- ❖ LLVM AddressSanitizer
  - Compiler-based code instrumentation
- ❖ Valgrind Memcheck
  - Binary instrumentation with the need of source code
- ❖ Linux Kernel Memory Leak Detector
  - Dynamic garbage collection-based algorithm
- ❖ And many others

However, detecting leaks in OS kernels is still challenging

The Linux kernel memory leak detector is a dynamic detection tool

- ➢ Cannot cover kernel code that is not executed at runtime

Existing static detection techniques are mainly developed for user applications

- ➢ Cannot be directly applied to OS kernels, which are much more complicated, e.g., 25 million source lines in Linux

Our focus: memory leaks on early-exit paths

Program paths designed to exit from kernel routines *as early as possible*

```
/* mm/mempool.c */
int mempool_resize(mempool_t *pool, int new_min_nr) {
    …
    spin_lock_irqsave(&pool->lock, flags);
    if (new_min_nr <= pool->min_nr) {
        spin_unlock_irqrestore(&pool->lock, flags);
        kfree(new_elements);
        return 0;
    }
    …
    return 0;
}
```

MLEE: detecting memory leaks on early-exit paths

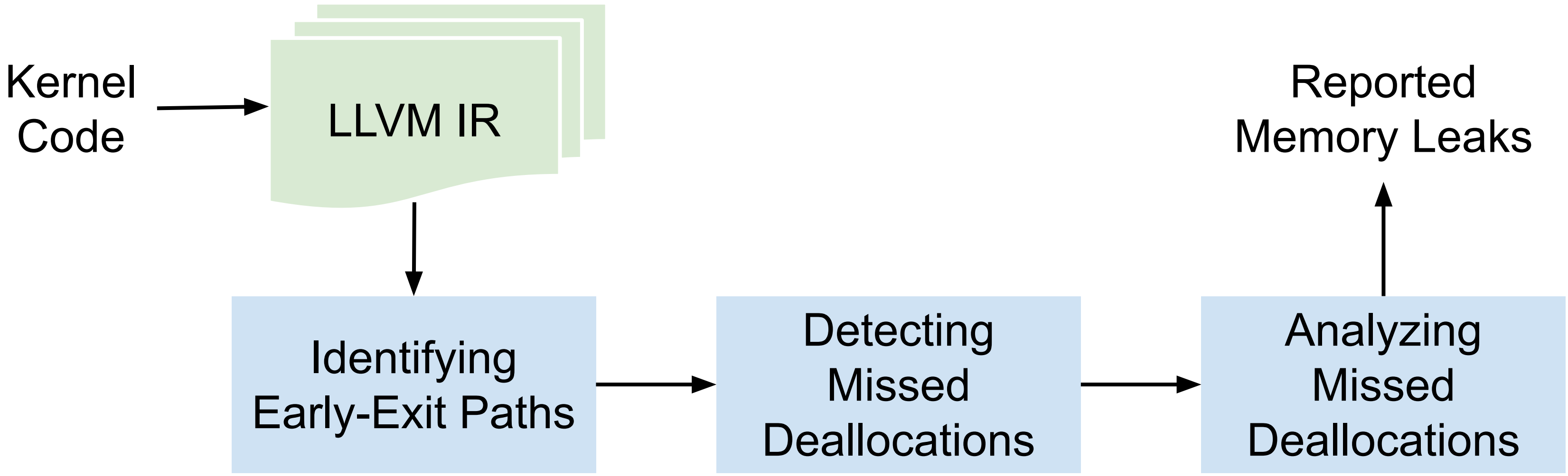
Idea

★ *Inconsistent presences* of memory deallocations on different early-exit paths and normal paths → potential memory leaks

Approach

★ *Cross-checking* presences of memory deallocations on different early-exit paths and normal paths

How MLEE works?



Challenge 1: how to identify early-exit paths?

Problem: the diverse semantics and usage scenarios of early-exit paths in OS kernels make it difficult to precisely identify early-exit paths

Our solution:

- Identifying the *unique* program statements of an early-exit path
  - **Early-exit branch:** a particular conditional branch that leads to an early-exit path

Challenge 2: how to analyze memory deallocations?

Problem: a missed memory deallocation may be *not needed* on a specific early-exit path

Our solution:

- ❑ Creating effective static analyses to ensure that the freed object is *live*, *valid* and *not used* on the early-exit path
- ❑ Heuristically inferring whether the missed memory deallocation is necessary

Experiments

MLEE is implemented as an LLVM tool with multiple analysis passes

- Based on LLVM 8.0.0

Applying MLEE to Linux (version 5.0) to detect memory leaks

- Compiling Linux to LLVM IR with the “allyes” option
- Detection time: around half an hour

Analysis results

Kernel routines with early-exit paths	121829
Kernel routines with memory deallocations	14540
Kernel routines with both early-exit paths and deallocations	7685
Early-exit branches with missed memory deallocations	126

120 new memory leaks are confirmed

- 87 (74.2%) have been fixed using our patches
- 16 (13.3%) have been fixed using others’ patches
- 15 (12.5%) have been confirmed and we are working on the final patches

drivers/	60%
sound/	18%
fs/	17%
others	5%

A memory leak found by MLEE

```
/* drivers/net/ethernet/mellanox/mlx4/en_rx.c */
int mlx4_en_config_rss_steer(struct mlx4_en_priv *priv)
{
    …
    err = mlx4_qp_alloc(mdev->dev, priv->base_qpn, ...);
    if (err) {
        en_err(priv, "Failed to allocate RSS ...\n");
        goto rss_err; // rss_map->indir_qp is leaked
    }                 // on this early-exit path.
    …
    kfree(rss_map->indir_qp);
    rss_map->indir_qp = NULL;
rss_err:
    …
    return err;
}
```

Limitations of MLEE

False negatives

- ➔ MLEE mainly focuses on memory leaks related to early-exit paths

False positives (18%)

- ➔ The memory object is deallocated by another kernel thread (8%)
- ➔ The memory object is deallocated in a callback routine (18%)
- ➔ Missed memory deallocations are not required (74%)

Summary

- ❖ Memory leaks in critical system software can cause serious system issues
- ❖ To detect memory leaks in OS kernels, we develop and implement MLEE
  - ✓ MLEE focuses on memory leaks on *early-exit paths*, which are often rarely tested in practice
  - ✓ After applying MLEE to the Linux kernel, we found **120 memory leaks** and most of them have been fixed
- ❖ With more tools like MLEE, more memory leaks can be found and fixed