Finding kernel memory leaks using kmemleak

Introduction

- Kmemleak is a Linux kernel feature designed to detect and report memory leaks in the kernel code. Memory leaks occur when a program allocates memory dynamically but fails to release it when it is no longer needed. This can result in a gradual loss of memory over time, eventually leading to system instability or even crashes.
- Kmemleak works by scanning the kernel memory periodically to detect any memory that has been allocated but not freed. When it detects a potential memory leak, it reports it to the system log along with a stack trace that shows where the memory was allocated. This information can help developers to identify and fix the source of the memory leak.
- Kmemleak is a useful tool for kernel developers and system administrators who want to ensure that their systems are stable and free of memory leaks. It can be enabled in the kernel configuration, and once enabled, it will run automatically in the background, periodically scanning the kernel memory for leaks.

Required kernel configs need to enable

```
CONFIG_HAVE_DEBUG_KMEMLEAK=y
CONFIG_DEBUG_KMEMLEAK=y
CONFIG_DEBUG_KMEMLEAK_TEST=m
```

Integrating Kmemleak tool for RPI-3

1) Create **kmemleak.cfg** file under **meta-cmf-raspberrypi/recipes-kernel/linux/files** with below kernel configs.
Note: Need to add this change in corresponding platform OEM layer.

```
CONFIG_HAVE_DEBUG_KMEMLEAK=y
CONFIG_DEBUG_KMEMLEAK=y
CONFIG_DEBUG_KMEMLEAK_TEST=m
```

2) Add **kmemleak.cfg** file in **meta-cmf-raspberrypi/recipes-kernel/linux/linux-raspberrypi_5.10.52.bb**
Note: Need to add this change in corresponding platform OEM layer.

Please find below diff file.
Ex:

```
diff --git a/recipes-kernel/linux/linux-raspberrypi_5.10.52.bb b/recipes-kernel/linux/linux-raspberrypi_5.10.52.
bb
index 2c7f19a..224a64a 100644
--- a/recipes-kernel/linux/linux-raspberrypi_5.10.52.bb
+++ b/recipes-kernel/linux/linux-raspberrypi_5.10.52.bb
@@ -11,6 +11,7 @@ FILESEXTRAPATHS_prepend := "${THISDIR}/files:"

 SRC_URI += " file://powersave.cfg \
              file://android-drivers.cfg \
+             file://kmemleak.cfg \
 "
```

3) Once build done with above changes, verify the kernel configs are enabled in final kernel boot config file.
Path of kernel boot config file: build-raspberrypi-rdk-mc/tmp/work/raspberrypi_rdk_mc-rdk-linux-gnueabi/linux-raspberrypi/1_5.10.52+gitAUTOINC+dbe03fa900_2697f74031-r0/image/boot/config-5.10.52-v7
Note: Need to verify in corresponding platform specific final kernel boot config file
Ex:

```
csures569@dvm-yocto3-docker-csures569:~/rpi-stack/build-raspberrypi-rdk-mc/tmp/work/raspberrypi_rdk_mc-rdk-
linux-gnueabi/linux-raspberrypi/1_5.10.52+gitAUTOINC+dbe03fa900_2697f74031-r0/image/boot$ cat config-5.10.52-
v7  | grep "KMEMLEAK"
CONFIG_HAVE_DEBUG_KMEMLEAK=y
CONFIG_DEBUG_KMEMLEAK=y
CONFIG_DEBUG_KMEMLEAK_MEM_POOL_SIZE=16000
CONFIG_DEBUG_KMEMLEAK_TEST=m
# CONFIG_DEBUG_KMEMLEAK_DEFAULT_OFF is not set
CONFIG_DEBUG_KMEMLEAK_AUTO_SCAN=y
```

4) On flashing the image with above changes , verify the kmemleak enabled in the device, it will create a **kmemleak folder** under **/sys/modules/** and **kmemleak file** under **/sys/kernel/debug/**
Ex:

```
root@raspberrypi-rdk-mc:/sys/module# ls -l | grep "kmemleak"
drwxr-xr-x    3 root     root             0 Mar  5 09:49 kmemleak

root@raspberrypi-rdk-mc:~# ls /sys/kernel/debug/kmemleak
/sys/kernel/debug/kmemleak
```

5) Once enabled kmemleak we will get kernel memory leaks in **/sys/kernel/debug/kmemleak file** if leaks are reported.

Testing kmemleak tool with kmemleak-test module in RPI-3

kmemleak-test is a test module available as part of kmemleak. By default, this module is not compiled in RPI3 and so there is no kmemleak-test.ko module. If needed, we need to enable to build as a module by following the above mentioned kernel configurations.
Path of kmemleak-test module source : build-raspberrypi-rdk-mc/tmp/work-shared/raspberrypi-rdk-mc/kernel-source/samples/kmemleak/kmemleak-test.c
kmemleak-test.koblocked URL

Note: Need to generate **kmemleak-test.ko** module file from corresponding platform specific kernel source code.

Step 1 : Copy kmemleak-test.ko module to this path **/lib/modules/5.10.52-v7/** in box .
Step 2 : load the module to kernel space with insmod command.
Ex:

```
root@raspberrypi-rdk-mc:~# insmod /lib/modules/5.10.52-v7/kmemleak-test.ko
root@raspberrypi-rdk-mc:~#
```

Step 3 : verify whether module is loaded or not
Ex: lsmod

```
root@raspberrypi-rdk-mc:~# lsmod

Module                Size      Used by

kmemleak_test    16384        0 // kmemleak test module loaded.
ip6t_REJECT         16384       2
nf_reject_ipv6      16384        1 ip6t_REJECT
ip6table_nat        16384        1
br_netfilter           32768       0
xt_state                16384       0
```

Step 4: Perform a scan
Ex: echo scan > /sys/kernel/debug/kmemleak

```
root@raspberrypi-rdk-mc:~# echo scan > /sys/kernel/debug/kmemleak
root@raspberrypi-rdk-mc:~#
```

Step 5: Collect memory leak data

Ex: cat /sys/kernel/debug/kmemleak

kmemleak_report...test_module.txt

kmemleak-test.ko

Note : In kmemleak-test module (**kmemleak-test.c**) the kernel memory leaks functionality present in the **kmemleak_test_init** function and **kmemleak_test_exit** function, so memory leaks will be detect when module loaded into kernel space and removed from kernel space.

Testing memory leaks from your kernel code

Suppose that you have a kernel module code which you want to test for kernel memory leaks. Make sure that kmemleak is enabled as mentioned above. Then, run the functionality of your kernel module by invoking it. Any memory leaks present in the kernel driver code will be reported in **/sys/kernel/debug/kmemleak.**

Memory leaks from module_init and module_exit functions get reported when the driver is inserted or removed. Inorder to test the memory leaks present in other functions , we need to manually invoke those functions.

Limitations and Drawbacks

While kmemleak can be a useful tool for detecting and reporting memory leaks in the Linux kernel, there are several limitations and drawbacks to consider: The main drawback is the reduced performance of memory allocation and freeing. To avoid other penalties, the memory scanning is only performed when the /sys/kernel/debug/kmemleak file is read. Anyway, this tool is intended for debugging purposes where the performance might not be the most important requirement.

1. False positives: Kmemleak may report potential memory leaks that are actually false positives, leading to unnecessary investigation and debugging.
2. Overhead: Kmemleak can increase the overhead of the kernel, particularly during the initial scanning phase. This can impact system performance and resource utilization.
3. Incomplete coverage: Kmemleak may not be able to detect all memory leaks, particularly those that occur in modules or code that are not regularly scanned.
4. Debugging complexity: Debugging memory leaks can be a complex and time-consuming process, particularly in large and complex kernel codebases.

Ref: https://static.lwn.net/kerneldoc/dev-tools/kmemleak.html