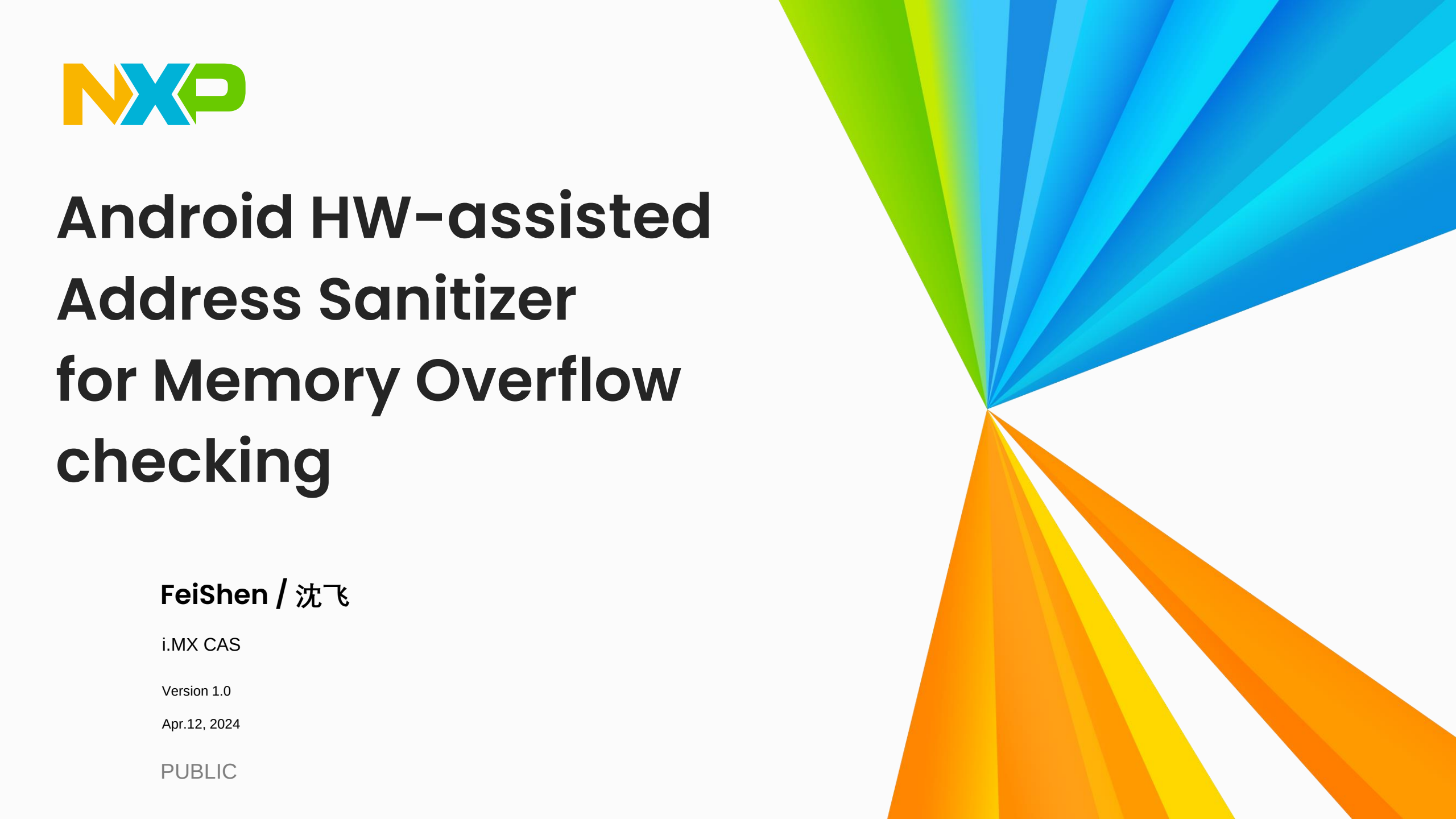


Android HW-assisted Address Sanitizer for Memory Overflow checking

FeiShen / 沈飞

i.MX CAS

Version 1.0

Apr.12, 2024

Agenda

- Introduction of HW-assisted Address Sanitizer (HWASan)
- Enable HWASan for NXP Android-13.0.0_2.3.0_auto BSP
- Example: WiFi-HAL Memory Overflow issue hunting
- Appendix
  - Kernel Address-Sanitizer

Introduction of HW-assisted Address Sanitizer (HWASan)

Hardware-assisted AddressSanitizer (HWASan) is a memory error detection tool similar to AddressSanitizer. HWASan uses a lot less RAM compared to ASan, which makes it suitable for whole system sanitization.

HWASan is based on the memory tagging approach, where a small random tag value is associated both with pointers and with ranges of memory addresses. For a memory access to be valid, the pointer and memory tags have to match.

HWASan is only available on **Android 10 and higher**, and only on **AArch64** hardware.

Enable HWASan for NXP Android-13.0.0_2.3.0_auto BSP (1)

Android-13.0.0_2.3.0_auto BSP is the latest Android-Auto BSP for i.MX (up to Date of this PPT).

It can be download from NXP public website:

https://www.nxp.com/design/design-center/software/embedded-software/i-mx-software/android-automotive-os-for-i-mx-applications-processors:ANDROID-AUTO

Releases

Android Automotive BSP Current Release

Release and Documentation	Build Sources	Supported Platforms/Binary Demo Files
Android Automotive 13.0.0_2.3.0 (Linux 6.1.36_2.1.0 BSP) Release Date: Jan 2024 Documentation • Android Auto Release Notes • i.MX Android Extended Codec Release Notes • Android Auto Quick Start Guide • Android Auto User's Guide • i.MX Graphics User's Guide • i.MX Android Security User's Guide • i.MX TensorFlow Lite on Android User's Guide	• See README for release intructions. • Install Source Package	• i.MX 8QuadMax and i.MX 8QuadXPlus (B0, C0) MEK – demo images with EVS in Arm Cortex-M4 core • i.MX 8QuadMax and i.MX 8QuadXPlus (B0, C0) MEK – demo images with EVS in Arm Cortex-A core

Enable HWASan for NXP Android-13.0.0_2.3.0_auto BSP （2)

Apply following patch to enable HWASan:

```
diff --git a/imx8q/mek_8q/BoardConfig.mk b/imx8q/mek_8q/BoardConfig.mk
index cbab0ea8..97adceb7 100644
--- a/imx8q/mek_8q/BoardConfig.mk
+++ b/imx8q/mek_8q/BoardConfig.mk
@@ -284,3 +287,6 @@ ifeq ($(PRODUCT_IMX_CAR),true)
 BOARD_HAVE_IMX_EVS := true
 endif

+# ----HWSan-------
+SANITIZE_TARGET=hwaddress
+
```

Example: WiFi-HAL Memory Overflow issue hunting （1)

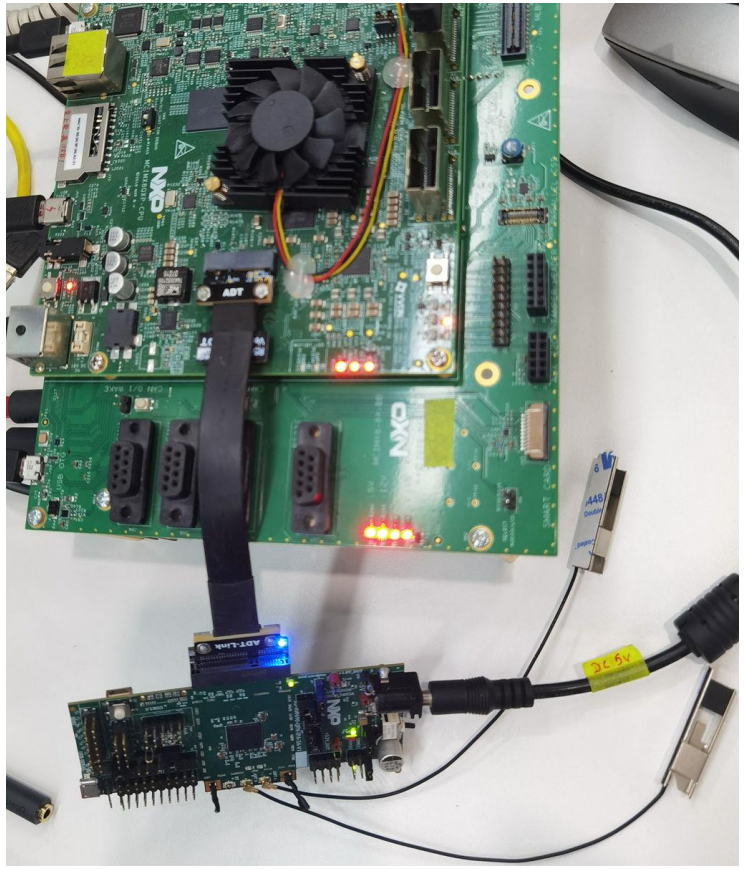
Test environment:

SW: Android-13.0.0_2.3.0_car2, pre-built image.

HW: 88W9098 WiFi/BT EVK (PCIe) + i.MX8QXP EVK.

Run 88W9098 WiFi/BT on i.MX8QXP EVK, after ~2 hours, got memory leakage.

```
mek_8q:/ #
[ 8209.899167][  T812] binder_alloc: 3436: pid 674 spamming oneway? 414 buffers allocated for a total size of 506736
[ 8212.897736][  T812] binder_alloc: 3436: pid 674 spamming oneway? 415 buffers allocated for a total size of 507968
[ 8212.909072][  T812] binder_alloc: 3436: pid 674 spamming oneway? 416 buffers allocated for a total size of 509184
[ 8252.048828][  T812] binder_alloc: 3436: pid 674 spamming oneway? 417 buffers allocated for a total size of 510416
[ 8252.060048][  T812] binder_alloc: 3436: pid 674 spamming oneway? 418 buffers allocated for a total size of 511632
[ 8255.061453][  T812] binder_alloc: 3436: pid 674 spamming oneway? 419 buffers allocated for a total size of 512864
[ 8255.072913][  T812] binder_alloc: 3436: pid 674 spamming oneway? 420 buffers allocated for a total size of 514080
[ 8273.132561][  T812] binder_alloc: 3436: pid 674 spamming oneway? 421 buffers allocated for a total size of 515312
[ 8273.143878][  T812] binder_alloc: 3436: pid 674 spamming oneway? 422 buffers allocated for a total size of 516528
[ 8276.146818][  T812] binder_alloc: 3436: pid 674 spamming oneway? 423 buffers allocated for a total size of 517760
[ 8276.158312][  T812] binder_alloc: 3436: pid 674 spamming oneway? 424 buffers allocated for a total size of 518976
[ 8468.858528][  T812] binder: cannot allocate buffer: no space left
[ 8468.858571][  T812] binder: 674:812 transaction async to 3436:0 failed 1401993/29201/-28, size 1124-0 line 3450
[ 8468.886524][  T812] binder_alloc: 3436: pid 674 spamming oneway? 425 buffers allocated for a total size of 520192
[ 8471.874746][  T812] binder: cannot allocate buffer: no space left
[ 8471.874793][  T812] binder: 674:812 transaction async to 3436:0 failed 1402371/29201/-28, size 1124-0 line 3450
[ 8471.892775][  T812] binder: cannot allocate buffer: no space left
[ 8471.892813][  T812] binder: 674:812 transaction async to 3436:0 failed 1402376/29201/-28, size 1112-0 line 3450
[ 8514.031241][  T812] binder: cannot allocate buffer: no space left
[ 8514.031284][  T812] binder: 674:812 transaction async to 3436:0 failed 1408166/29201/-28, size 1124-0 line 3450
[ 8514.049469][  T812] binder: cannot allocate buffer: no space left
```



Example: WiFi-HAL Memory Overflow issue hunting （2)

Enabled HWASan, re-build Android-13.0.0_2.3.0_auto BSP, run again,

Got detail HWAddressSanitizer report: “heap-buffer-overflow”.

--Reason> "Empty or null ScanResult list" ->
--Then> "Attempt to retrieve OsuProviders with invalid scanResult List" ->
--Result> "heap-buffer-overflow"

```
04-11 11:41:26.899   715  4523 W ScanResultUtil: Empty or null ScanResult list
04-11 11:41:26.901   715  4523 W WifiService: Attempt to retrieve OsuProviders with invalid scanResult List
04-11 11:41:26.901   715  1042 E WifiVendorHal: getUsableChannels(1.4133) failed {.code = ERROR_NOT_SUPPORTED, .description = }
04-11 11:41:26.930   715  1042 E ApConfigUtil: Malformed channel value detected: java.lang.NumberFormatException: For input string: ""
04-11 11:41:26.934   715  1042 E WifiVendorHal: getUsableChannels(1.4133) failed {.code = ERROR_NOT_SUPPORTED, .description = }
04-11 11:41:26.951   715  1042 E WifiVendorHal: getUsableChannels(1.4133) failed {.code = ERROR_NOT_SUPPORTED, .description = }
04-11 11:41:26.979   715  1042 E ApConfigUtil: Malformed channel value detected: java.lang.NumberFormatException: For input string: ""
04-11 11:41:26.984  5503  5503 I wpa_supplicant: the nl80211 driver cmd is SETSUSPENDMODE 0
04-11 11:41:26.984  5503  5503 I wpa_supplicant: the nl80211 driver cmd len is 16
04-11 11:41:26.995   715  1729 I WifiService: startScan uid=1001000
04-11 11:41:27.006   715  1135 I WifiService: startScan uid=1001000
04-11 11:41:27.007   715  1731 I WifiService: startScan uid=1001000
04-11 11:41:27.008   715  1483 I WifiService: startScan uid=1001000
04-11 11:41:27.036   715  1483 W ScanResultUtil: Empty or null ScanResult list
04-11 11:41:27.037   715  1483 W WifiService: Attempt to retrieve WifiConfiguration with invalid scanResult List
04-11 11:41:27.043  4942  4942 I android.hardware.wifi@1.0-service: ==4942==ERROR: HWAddressSanitizer: tag-mismatch on address 0x004826bd017
04-11 11:41:27.043  4942  4942 I android.hardware.wifi@1.0-service: WRITE of size 8 at 0x004826bd0178 tags: d7/08(d7) (ptr/mem) in thread T0
04-11 11:41:27.062   715  1483 W ScanResultUtil: Empty or null ScanResult list
04-11 11:41:27.062   715  1483 E WifiService: Attempt to retrieve passpoint with invalid scanResult List
04-11 11:41:27.065   715  1483 W ScanResultUtil: Empty or null ScanResult list
04-11 11:41:27.065   715  1483 W WifiService: Attempt to retrieve OsuProviders with invalid scanResult List scanResult List
01-01 00:11:01.554   624   624 F wlan    : wlan0 START SCAN
04-11 11:41:27.077  4942  4942 I android.hardware.wifi@1.0-service:     #0 0x7c80e9be1c  (/apex/com.android.runtime/lib64/bionic/libclang_rt
04-11 11:41:27.079  4942  4942 I android.hardware.wifi@1.0-service:     #1 0x7c81ea5f24  (/vendor/lib64/libwifi-hal.so+0x1af24) (BuildId: ac
04-11 11:41:27.080  4942  4942 I android.hardware.wifi@1.0-service:     #2 0x7c81ea51a8  (/vendor/lib64/libwifi-hal.so+0x1a1a8) (BuildId: ac
04-11 11:41:27.081  4942  4942 I android.hardware.wifi@1.0-service:     #3 0x7c7d5ac410  (/apex/com.android.vndk.v33/lib64/libnl.so+0x22410)
04-11 11:41:27.082  4942  4942 I android.hardware.wifi@1.0-service:     #4 0x7c7d5ac608  (/apex/com.android.vndk.v33/lib64/libnl.so+0x22608)
04-11 11:41:27.083  4942  4942 I android.hardware.wifi@1.0-service:     #5 0x7c81ea4fb0  (/vendor/lib64/libwifi-hal.so+0x19fb0) (BuildId: ac
04-11 11:41:27.085  4942  4942 I android.hardware.wifi@1.0-service:     #6 0x7c81ea57a0  (/vendor/lib64/libwifi-hal.so+0x1a7a0) (BuildId: ac
04-11 11:41:27.086  4942  4942 I android.hardware.wifi@1.0-service:     #7 0x5c26c65f5c  (/vendor/bin/hw/android.hardware.wifi@1.0-service+0
04-11 11:41:27.088  4942  4942 I android.hardware.wifi@1.0-service:     #8 0x5c26c96c88  (/vendor/bin/hw/android.hardware.wifi@1.0-service+0
04-11 11:41:27.100  4942  4942 I android.hardware.wifi@1.0-service:     #9 0x5c26c96848  (/vendor/bin/hw/android.hardware.wifi@1.0-service+0
04-11 11:41:27.100  4942  4942 I android.hardware.wifi@1.0-service:     #10 0x5c26c96b38  (/vendor/bin/hw/android.hardware.wifi@1.0-service+
04-11 11:41:27.101  4942  4942 I android.hardware.wifi@1.0-service:     #11 0x7c795b9510  (/vendor/lib64/android.hardware.wifi@1.6.so+0x1345
04-11 11:41:27.101  4942  4942 I android.hardware.wifi@1.0-service:     #12 0x7c795bb010  (/vendor/lib64/android.hardware.wifi@1.6.so+0x1360
04-11 11:41:27.101  4942  4942 I android.hardware.wifi@1.0-service:     #13 0x7c7b752b14  (/apex/com.android.vndk.v33/lib64/libhidlbase.so+0
04-11 11:41:27.101  4942  4942 I android.hardware.wifi@1.0-service:     #14 0x7c7b759b84  (/apex/com.android.vndk.v33/lib64/libhidlbase.so+0
04-11 11:41:27.101  4942  4942 I android.hardware.wifi@1.0-service:     #15 0x7c7b7591d0  (/apex/com.android.vndk.v33/lib64/libhidlbase.so+0
04-11 11:41:27.101  4942  4942 I android.hardware.wifi@1.0-service:     #16 0x7c7b75a40c  (/apex/com.android.vndk.v33/lib64/libhidlbase.so+0
04-11 11:41:27.101  4942  4942 I android.hardware.wifi@1.0-service:     #17 0x5c26c0c630  (/vendor/bin/hw/android.hardware.wifi@1.0-service+
04-11 11:41:27.101  4942  4942 I android.hardware.wifi@1.0-service:     #18 0x7c7ced4908  (/apex/com.android.runtime/lib64/bionic/libc.so+0x
04-11 11:41:27.102  4942  4942 I android.hardware.wifi@1.0-service:
04-11 11:41:27.102  4942  4942 I android.hardware.wifi@1.0-service: [0x004826bd0000,0x004826bd0180) is a small allocated heap chunk; size: 3
04-11 11:41:27.103  4942  4942 I android.hardware.wifi@1.0-service:
04-11 11:41:27.103  4942  4942 I android.hardware.wifi@1.0-service: Cause: heap-buffer-overflow
04-11 11:41:27.104  4942  4942 I android.hardware.wifi@1.0-service: 0x004826bd0178 is located 0 bytes to the right of 376-byte region [0x004
04-11 11:41:27.105  4942  4942 I android.hardware.wifi@1.0-service: allocated here:
```

Appendix

Kernel Address-Sanitizer (KASan)

Kernel Address-Sanitizer (KASan)

Similar to the LLVM-based sanitizers for userspace components, Android includes the Kernel Address Sanitizer (KASAN). KASAN is a combination of kernel and compile time modifications that result in an instrumented system that allows for simpler bug discovery and root cause analysis.

KASAN can detect many types of memory violations in the kernel. It can also detect out-of-bound reads and writes on stack, heap and global variables, and can detect use-after-free and double frees.

Similar to ASAN, KASAN uses a combination of memory-function instrumentation at compile time and shadow memory to track memory accesses at runtime.

Software Tag-Based KASAN, enabled with CONFIG_KASAN_SW_TAGS, can be used for both debugging and dogfood testing. This mode is only supported for **arm64**, but its moderate memory overhead allows using it for testing on memory-restricted devices with real workloads.

Hardware Tag-Based KASAN, enabled with CONFIG_KASAN_HW_TAGS, is the mode intended to be used as an in-field memory bug detector or as a security mitigation. This mode only works on **arm64** CPUs that support MTE (Memory Tagging Extension), but it has low memory and performance overheads and thus can be used in production.

Implementation

To compile a kernel with KASAN enabled, add the following build flags to your kernel build configuration:

```
CONFIG_KASAN=y
CONFIG_KASAN_HW_TAGS=y
```

nxp.com

PUBLIC