MTE Findings

Arm v9 introduces Arm Memory Tagging Extension (MTE), a hardware implementation of tagged memory.

At a high level, MTE tags each memory allocation/deallocation with additional metadata. It assigns a tag to a memory location, which then can be associated with pointers that reference that memory location. At runtime the CPU checks that the pointer and the metadata tags match on each load and store.

In Android 12 the kernel and userspace heap memory allocator can augment each allocation with metadata. This helps detect use-after-free and buffer-overflow bugs, which are the most common source of memory safety bugs in our codebases.

MTE operating modes

MTE has three operating modes:

- Synchronous mode (SYNC)
- Asynchronous mode (ASYNC)
- Asymmetric mode (ASYMM)

Check for support

Starting from Android 13, select devices have support for MTE. To check whether your device is running with MTE enabled, run the following command:

```
adb shell grep mte /proc/cpuinfo
```

If the result is `Features : [...] mte`, your device is running with MTE enabled.

Some devices don't enable MTE by default, but allow developers to reboot with MTE enabled. This is an experimental configuration that is not recommended for normal use as it might decrease device performance or stability, but can be useful for app development. To access this mode, navigate to Developer

Options > Memory Tagging Extension in your Settings App. If this option is not present, your device does not support enabling MTE this way.

Enable MTE

For a single device

For experimentation, app compatibility changes can be used to set the default value of `memtagMode` attribute for an application that does not specify any value in the manifest (or specifies `"default"`).

These can be found under System > Advanced > Developer options > App Compatibility Changes in the global setting menu.
Setting `NATIVE_MEMTAG_ASYNC` or `NATIVE_MEMTAG_SYNC` enables MTE for a particular application.

Alternatively, this can be set using the `am` command as follows:

- For SYNC mode: `$ adb shell am compat enable NATIVE_MEMTAG_SYNC my.app.name`
- For ASYNC mode: `$ adb shell am compat enable NATIVE_MEMTAG_ASYNC my.app.name`

In Gradle

You can enable MTE for all debug builds of your Gradle project by putting

(off|sync|async)

```
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_r
ules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MTE_test"
        tools:targetApi="31"
        android:memtagMode="sync"
        tools:ignore="MissingPrefix">
    </application>
```

into `app/src/AndroidManifest.xml`.

Crash report

Now in order to check the output of MTE. Let's run the following example:

```
extern "C" JNIEXPORT void JNICALL
Java_com_example_testapp_1mem_MainActivity_doNullDeref(
        JNIEnv *env,
        jobject /* this */) {
    char * volatile p = (char *)nullptr;
    p[42] = 1;
}
```

At the crash event, the **Logcat** window in Android Studio shows information about tombstone

```
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *
Build fingerprint: 'google/shiba/shiba:14/AP1A.240305.019.A1/
Revision: 'MP1.0'
ABI: 'arm64'
Timestamp: 2024-04-16 09:00:01.709096386+0300
Process uptime: 4s
Cmdline: com.example.testapp_mem
pid: 9292, tid: 9292, name: ple.testapp_mem  >>> com.example.
uid: 10328
tagged_addr_ctrl: 000000000007fff3 (PR_TAGGED_ADDR_ENABLE, PR
pac_enabled_keys: 000000000000000f (PR_PAC_APIAKEY, PR_PAC_AP
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x00000
Cause: null pointer dereference
    x0  09000073fe0fe210  x1  0000007fc51fdb88  x2  0000007fc
    x4  000000732e131190  x5  000000732e131190  x6  0000007fc
    x8  0000000000000001  x9  0000000000000000  x10 000000000
    x12 0000000000000008  x13 0000000000002015  x14 ffffffff
    x16 0000000000000001  x17 0000007276ef91e8  x18 000000753
    x20 0000000000000000  x21 030000738e109448  x22 000000718
```

```
    x24 0000007fc51fdbd0  x25 0a000073be0f5010  x26 000000000
    x28 0000007fc51fdaa0  x29 0000007fc51fdb90
    lr  000000727b351e34  sp  0000007fc51fda70  pc  000000727
87 total frames
backtrace:
      #00 pc 0000000000027200  /data/app/~~48Pyy084WfF9OYqbRG
      #01 pc 0000000000351e30  /apex/com.android.art/lib64/li
      #02 pc 000000000033b3a4  /apex/com.android.art/lib64/li
      #03 pc 0000000000511590  /apex/com.android.art/lib64/li
      #04 pc 0000000000490ffc  /apex/com.android.art/lib64/li
      #05 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #06 pc 0000000000001b50  /data/app/~~48Pyy084WfF9OYqbRG
      #07 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #08 pc 000000000051225c  /apex/com.android.art/lib64/li
      #09 pc 000000000049136c  /apex/com.android.art/lib64/li
      #10 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #11 pc 00000000000019ac  /data/app/~~48Pyy084WfF9OYqbRG
      #12 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #13 pc 000000000051225c  /apex/com.android.art/lib64/li
      #14 pc 000000000049136c  /apex/com.android.art/lib64/li
      #15 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #16 pc 00000000000018a0  /data/app/~~48Pyy084WfF9OYqbRG
      #17 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #18 pc 000000000051225c  /apex/com.android.art/lib64/li
      #19 pc 0000000000491eac  /apex/com.android.art/lib64/li
      #20 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #21 pc 0000000000208198  /system/framework/framework.ja
      #22 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #23 pc 000000000051225c  /apex/com.android.art/lib64/li
      #24 pc 000000000049136c  /apex/com.android.art/lib64/li
      #25 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #26 pc 0000000000304908  /data/app/~~48Pyy084WfF9OYqbRG
      #27 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #28 pc 000000000036dfe4  /apex/com.android.art/lib64/li
      #29 pc 0000000000351f68  /apex/com.android.art/lib64/li
      #30 pc 00000000005b98b0  /apex/com.android.art/lib64/li
      #31 pc 00000000002081ee  /system/framework/framework.ja
      #32 pc 00000000005b9854  /apex/com.android.art/lib64/li
```

```
      #33 pc 0000000000203234  /system/framework/framework.ja
      #34 pc 000000000033b680  /apex/com.android.art/lib64/li
      #35 pc 0000000000511714  /apex/com.android.art/lib64/li
      #36 pc 000000000049136c  /apex/com.android.art/lib64/li
      #37 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #38 pc 00000000001dbec8  /system/framework/framework.ja
      #39 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #40 pc 000000000051225c  /apex/com.android.art/lib64/li
      #41 pc 0000000000491eac  /apex/com.android.art/lib64/li
      #42 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #43 pc 00000000001b809c  /system/framework/framework.ja
      #44 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #45 pc 000000000051225c  /apex/com.android.art/lib64/li
      #46 pc 000000000049136c  /apex/com.android.art/lib64/li
      #47 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #48 pc 00000000001b7ee4  /system/framework/framework.ja
      #49 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #50 pc 000000000051225c  /apex/com.android.art/lib64/li
      #51 pc 0000000000490ffc  /apex/com.android.art/lib64/li
      #52 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #53 pc 00000000001dbe60  /system/framework/framework.ja
      #54 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #55 pc 000000000051225c  /apex/com.android.art/lib64/li
      #56 pc 000000000049136c  /apex/com.android.art/lib64/li
      #57 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #58 pc 00000000001dc69c  /system/framework/framework.ja
      #59 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #60 pc 000000000051225c  /apex/com.android.art/lib64/li
      #61 pc 000000000049136c  /apex/com.android.art/lib64/li
      #62 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #63 pc 00000000001ccb9c  /system/framework/framework.ja
      #64 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #65 pc 000000000036dfe4  /apex/com.android.art/lib64/li
      #66 pc 0000000000351f68  /apex/com.android.art/lib64/li
      #67 pc 000000000033b680  /apex/com.android.art/lib64/li
      #68 pc 000000000037cb18  /apex/com.android.art/lib64/li
      #69 pc 000000000037c4f4  /apex/com.android.art/lib64/li
      #70 pc 0000000000351e30  /apex/com.android.art/lib64/li
```

```
      #71 pc 000000000033b3a4  /apex/com.android.art/lib64/li
      #72 pc 0000000000511590  /apex/com.android.art/lib64/li
      #73 pc 0000000000490ffc  /apex/com.android.art/lib64/li
      #74 pc 00000000003545d8  /apex/com.android.art/lib64/li
      #75 pc 000000000050ea0c  /system/framework/framework.ja
      #76 pc 000000000036e6ec  /apex/com.android.art/lib64/li
      #77 pc 000000000036dfe4  /apex/com.android.art/lib64/li
      #78 pc 0000000000351f68  /apex/com.android.art/lib64/li
      #79 pc 0000000000b88a3c  /data/misc/apexdata/com.androi
      #80 pc 000000000033b680  /apex/com.android.art/lib64/li
      #81 pc 00000000004e2a90  /apex/com.android.art/lib64/li
      #82 pc 000000000057aa68  /apex/com.android.art/lib64/li
      #83 pc 00000000000d8448  /system/lib64/libandroid_runti
      #84 pc 00000000000e450c  /system/lib64/libandroid_runti
      #85 pc 00000000000025b8  /system/bin/app_process64 (mai
      #86 pc 0000000000056e18  /apex/com.android.runtime/lib6
Destroyed 0 sockets, proto=IPPROTO_TCP, family=AF_INET, state
Destroyed 0 sockets, proto=IPPROTO_TCP, family=AF_INET6, stat
Destroyed live tcp sockets for uids={10165} in 5ms
add tag=data_app_native_crash isTagEnabled=true flags=0x6
  Force finishing activity com.example.testapp_mem/.MainActiv
Tombstone written to: tombstone_18
```

Tombstones

When your application crashes, a basic crash dump is written to the **Logcat** window in Android Studio. More detailed information is written to a tombstone file, located in the `/data/tombstones/` directory. This tombstone file contains detailed data about the crashed process, including the following:

To retrieve the tombstone file, use `adb bugreport` to capture the bug report. After unzipping the bug report file, the tombstone files are in the folder `/FS/data/tombstones/`.

Interpreting crash report

Understanding MTE reports | Android Open Source Project

SIGSEGV crashes with code 9 (SEGV_MTESERR) or code 8 (SEGV_MTEAERR) are Memory Tagging faults.

Memory Tagging Extension (MTE) is an

https://source.android.com/docs/security/test/memory-safety/mte-reports

Documentation – Arm Developer

https://developer.arm.com/documentation/108035/0100/MTE-bug-reports/Interpreting-the-bug-report?lang=en

References

- https://developer.arm.com/documentation/108035/0100/How-does-MTE-work-?lang=en
- https://developer.android.com/ndk/guides/arm-mtehttps://source.android.com/docs/security/test/memory-safety/arm-mte