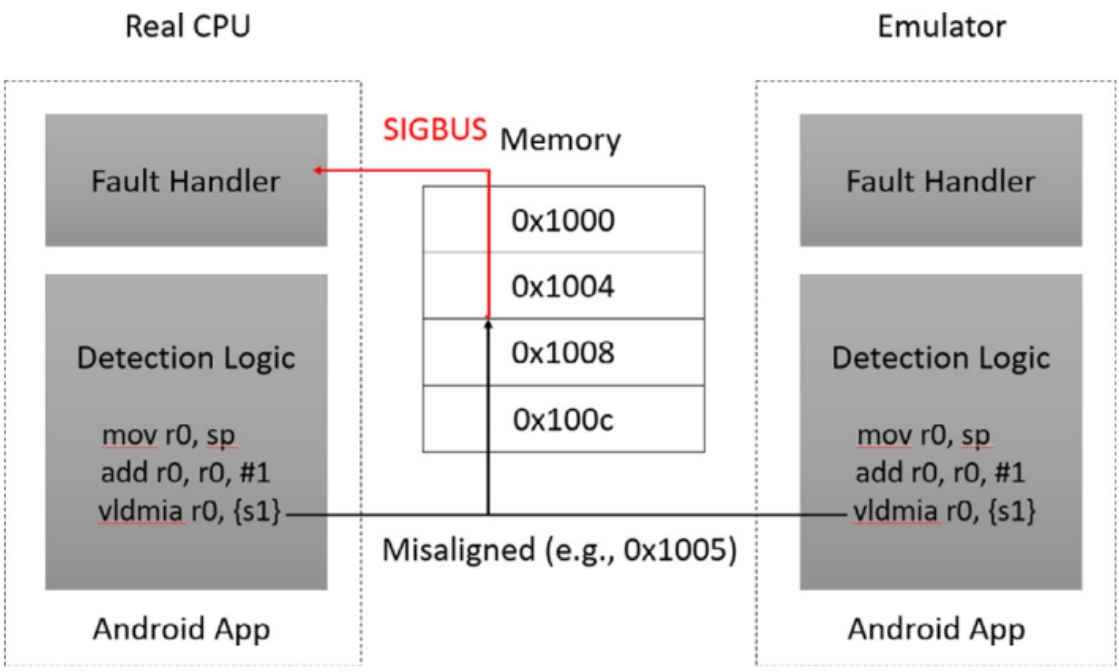
一、前言

本篇介绍一种基于字节对齐的*Anti-Unidbg*方法。

二、描述

《重新思考大型软件的反模拟器》一文中，提出了一种高效的反模拟器方法。



> ARM 中 SIMD(Single Instruction Multiple Data) 指令不支持对内存的非对齐访问，但QEMU等模拟器不存在这一限制。

QEMU以及基于它的Unicorn不存在字节对齐并不奇怪，事实上，在我认知里Unicorn除了页对齐并无其他对齐要求，至于前半句，ARM中SIMD是否支持对内存的非对齐访问，我有点把握不住，没找到足够多的标准资料，所以在这里引用一些相关的资料，读者可以做进一步的思考

Using the Stack in AArch32 and AArch64 - Processors blog - Processors - Arm Community

> For AArch32 (ARM or Thumb), `sp` must be at least 4-byte aligned *at all times*. As long as you only push and pop whole registers, this restriction will never be broken.
>
> For AArch64, `sp` must be 16-byte aligned *whenever it is used to access memory*. This is enforced by AArch64 hardware.

闲聊对齐异常(alignment fault) (happyseeker.github.io)

> ARM其"单指令"操作支持非对齐，但"群指令"操作(SIMD)则不支持(必须对齐访问)。

ARM非对齐访问和Alignment Fault - 者旨於陽 - 博客园 (cnblogs.com)

> Linux内核只针对arm架构实现了非对齐访问处理机制，主要是针对LDR, STR, LDRD, LDRD, STRD, LDM, STM, LDRH, STRH指令的非对齐访问进行处理。对于arm64架构，因为ARMv8架构CPU可以处理所有LDR/STR类内存访问指令的非对齐访问，因此没有实现该机制。这样就会导致如果在arm64架构上运行64位Linux内核，而用户态为32位应用程序时，如果发生非对齐访问，则会触发异常。

可以发现资料比较杂乱，也没有一个固定统一的说法，因此这里只讨论在我的测试机上成功的情况，或许是一种潜在的、可能的Anti-模拟值的方式。

首先是gradle中强制arm32，读者也可以去适配arm64的情况，demo弄的简单一些比较好。

```
defaultConfig {
    applicationId "com.example.testasm"
    minSdk 21
    targetSdk 31
    versionCode 1
    versionName "1.0"

    testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    externalNativeBuild {
        cmake {
            cppFlags ''
        }
    }
    ndk {
        abiFilters "armeabi-v7a"
    }

}
```

CMakeLists.txt 中添加对汇编的支持，因为我们要写内联汇编

```
set(can_use_assembler TRUE)
enable_language(ASM)
```

native-lib.cpp

```
#include <jni.h>
#include <string>


void trap()
{
    __asm(
    "ADD SP, SP, #1\t\n"
    "STMFD SP!,{R0-R3}\t\n"
    "LDMFD SP!,{R0-R3}\t\n"
    "SUB SP, SP, #1\t\n"
    );
}



extern "C"
JNIEXPORT jint JNICALL
Java_com_example_testasm_MainActivity_a(JNIEnv *env, jobject thiz) {
    trap();
    return 0;
}
```

通过内联汇编向SP+1的位置做读写，在真机上导致App崩溃，而Unidbg模拟执行不会出任何问题。

当然，我们并不希望App崩溃，所以需要在代码中实现自己的信号处理函数，当此处发生异常时，信号处理函数接收信号并做出某种处理，因为Unidbg中程序不会异常，所以也不会走到信号处理函数，这里面可以设计形成差异。

内联汇编学习资料

ARM GCC Inline Assembler Cookbook (ethernut.de)

可以通过宏定义识别arm32 or arm64，编写不同ABI下的内联汇编。

CPU 功能 | Android NDK | Android Developers

三、尾声

无。