#include <jni.h>

#include <string>

#include <chrono>

#include <random>

#include <unistd.h>

#include <signal.h>

#include <setjmp.h>

#include <android/log.h>

static jmp\_buf jump\_buffer;

void handle\_sigsegv(int signal) {

longjmp(jump\_buffer, 1);

}

jstring allocateAndMeasureAccessTime(JNIEnv \*env) {

const size\_t bufferSize = 65536; // 64KB

char\* buffer = new char[bufferSize];

std::default\_random\_engine generator;

std::uniform\_int\_distribution<int> distribution(0, bufferSize - 1);

struct sigaction sa;

sa.sa\_flags = SA\_SIGINFO;

sigemptyset(&sa.sa\_mask);

sa.sa\_handler = handle\_sigsegv;

sigaction(SIGSEGV, &sa, NULL);

std::string result;

// Simulate valid access

auto startTime = std::chrono::high\_resolution\_clock::now();

buffer[distribution(generator)] = 'a'; // Random access within the buffer

auto endTime = std::chrono::high\_resolution\_clock::now();

std::chrono::duration<double, std::nano> accessTime = endTime - startTime;

result = "Random valid access time: " + std::to\_string(accessTime.count()) + " nanoseconds\n";

// Simulate invalid access

if (setjmp(jump\_buffer) == 0) {

startTime = std::chrono::high\_resolution\_clock::now();

volatile char data = \*(buffer + bufferSize + distribution(generator)); // Beyond buffer

endTime = std::chrono::high\_resolution\_clock::now(); // Not expected to be executed

} else {

endTime = std::chrono::high\_resolution\_clock::now();

}

accessTime = endTime - startTime;

result += "Random invalid access time: " + std::to\_string(accessTime.count()) + " nanoseconds\n";

//delete[] buffer;

\_\_android\_log\_print(*ANDROID\_LOG\_INFO*, "MTEBenchmarking", "%s", result.c\_str());

return env->NewStringUTF(result.c\_str());

}

extern "C"

JNIEXPORT jstring JNICALL

Java\_com\_example\_mtestudy\_1benchmarking\_MainActivity\_stringFromJNI(JNIEnv \*env, jobject thiz) {

return allocateAndMeasureAccessTime(env);

}