**네이티브 라이브러리**

개발자가 작성한 다음 특정 컴퓨터가 아키텍처용으로 컴파일한 코드, 대부분 C와 C++로 작성된 코드

기존의 C/C++로 개발한 코드가 안드로이드 플랫폼에서 동작하는 바이너리로 빌드

🡺 그래픽 라이브러리와 같이 수학적으로 집약적이거나 시간에 민감한 작업을 위한 것

DEX 바이트코드 🡪 Java로 디컴파일 가능

컴파일된 코드 🡪 어셈블리로 분석

**JNI(Java Native Interface)**

개발자가 네이티브 코드로 구현되는 Java 메소드를 선언할 수 있음

Android NDK 🡪 JNI를 기반으로 하는 Android 전용 도구

Android 기본 라이브러리 🡪 .so ELF 파일 형식의 공유 객체 라이브러리로 APK에 포함

애플리케이션이 라이브러리를 메모리에 로드하기 위한 API 호출

1. System.loadLibrary(“calc”)

2. System.load(“lib/armeabi/libclac.so”)

두 API 중 하나가 실행되면 JNI\_OnLoad 네이티브 라이브러리 함수도 실행

Java 선언 네이티브 메서드 호출 🡪 네이티브 라이브러리의 페어링된 네이티브 함수가 실행

ex) public native String doThingsInNativeLibrary(int var0);

**페어링 하는 법**

1. JNI 네이티브 메소드 이름 확인을 사용한 동적 연결

2. RegisterNativesAPI 호출을 이용한 정적 링크

**동적 연결**

동적 연결을 위한 이름 지정

con.android.interesting.Stuff에서의 연결

Java\_com\_android\_interesting\_Stuff\_doThingsInNativeLibrary

네이티브 라이브러리에 해당 이름의 함수가 없으면 정적 수행 필요

**정적 연결**

개발자가 사양에 따라 기본 함수의 이름을 지정하기를 원하지 않거나,

지정할 수 없는 경우 정적 연결을 사용

jint RegisterNatives(JNIEnv \*env, jclass clazz, const JNINativeMethod \*methods, jint nMethods);

typedef struct {

char \*name;

char \*signature;

void \*fnPtr;

} JNINativeMethod;

**리버스 엔지니어링 – 정적 연결**

Java 선언 네이티브 메서드가 호출될 때 네이티브 라이브러리의 서브 루틴을 결정하기 위해 RegisterNative로 전달되는 JNINative 메서드 구조를 찾음

🡺 JNINatvie 메서드 구조에는 Java에서 선언한 네이티브 메서드 이름의 문자열과 메서드 signature의 문자열이 필요하므로 네이티브 라이브러리에서 찾을 수 있음

**Method Signature**

JNINative method 구조에는 method signature이 필요

🡺 인수 유형과 반환하는 인수 유형

Z 🡪 Boolean / B 🡪 Byte / C 🡪 Char / S 🡪 Short / I 🡪 Int / J 🡪 Long / F 🡪 Float

D 🡪 Double / L fully-qualified-class 🡪 fully-qualified-class / [ type 🡪 type[]

(arg-types) ret-type 🡪 method type / V 🡪 Void

실습 5 🡪 네이티브 메서드가 호출될 때 실행되는 네이티브 함수 식별

JNIEnv 🡺 JNI 함수에 대한 함수 포인터의 구조, Android 기본 라이브러리의 모든 JNI 기능은 JNIEnv\*를 첫 번째 인수로 사용