RQ1. What is the current state of real Android apps using native code?

Dataset

- 3682 apps from F-Droid (F-Droid_report.txt)
- 3549 apps from AndroZoo (2020.01 - 2022.09)

a. Native Library usage.

	F-Droid	AndroZoo		F-Droid	AndroZoo
Total App	3682	3549			
Has Native Method	808	2829	/Total App	21.9%	79.7%
Has .so File	799	826	/Total App	21.7%	23.3%
Has ELF in asset	39	769	/Total App	1.1%	21.7%
Has Encrypted zip	0	19	/Total App	-	0.5%
Has Native Activity	3	0	/Total App	0.1%	-
Total Native Method	127300	330477	/Has Native Method	157.5	116.8

Results of native library usage are presented in above Table. They indicate that 808 (i.e., 21.9%) of benign apps contain native method declarations and 799 (i.e., 21.7%) of benign apps contain at least one .so file. Regarding malware, 2829 (i.e., 79.7%) of apps contain native method declarations, and 826 (i.e., 23.3%) of apps contain at least one .so file. This means the .so file is probably downloaded at runtime or hidden under non-standard (i.e., assets) folder. We also analyzed these folder and found that 769 malware apps and 39 benign apps hide .so files in the assets folder by modifying the extension or compressing them. The most common wrong extension name we found are jar, png, zip and sdk. When decompressing zip files, we also found that some malicious apps encrypted these zip files. In addition, only a few apps used Native Activity components, so JNFuzz-Droid did not consider this situation.

b. cpu architecture

	F-Droid	AndroZoo		F-Droid	AndroZoo
Total .so File	799	826			
armeabi	211	588	/Total .so File	26.4%	71.2%
armeabi-v7a	729	709	/Total .so File	91.2%	85.8%
arm64-v8a	630	563	/Total .so File	78.8%	68.2%
x86	536	331	/Total .so File	67.1%	40.1%
x86_64	588	221	/Total .so File	73.6%	26.8%
mips	100	27	/Total .so File	12.5%	32.7%
mips 64	57	22	/Total .so File	7.1%	2.7%
other	3	21	/Total .so File	0.4%	2.5%

overall， ARM is the most popular CPU architecture for android. among them arm64-v8a and armeabi-v7a accounts for a significant proportion.

c. native code information

To better fuzz native code, we further investigate native code information, and the experimental results are categorized into three parts. The investigation results are shown in Table below.

	F-Droid	AndroZoo		F-Driod	AndroZoo
Part A: Static Native Method					
Has NM	808	2829			
Total NM	127300	330477	/Has Name Method	157.5	116.8
Static NM	75405	149815	/Total Name Method	59.2%	45.3%
Non-static NM	51895	180662	/Total Name Method	40.8%	54.7%
Part B: Parameter Types in Native Methods					
Total NM Par	399548	1589570	/Total Name Method	3.1	4.8
Simple Type	328842	920819	/Total Name Method Parameter	82.3%	57.9%
Interaction Type	4968	647500	/Total Name Method Parameter	1.2%	40.7%
Complex Type	65738	21251	/Total Name Method Parameter	16.5%	1.3%
Part C: Native Method Registration Ways					
Has .so File	799	826			
Total .so File	4592	3157	/Has .so File	5.7	3.8
Static Register	925	1957	/Total .so File	20.1%	62.0%
Dynamic Register	1186	1485	/Total .so File	25.8%	47.0%

In part A, static native methods account for 59.2% of malware and 45.3% of benign.

note: We divided them into 3 categories according to frequency and function of the parameter types: Simple type, Interactive type, and Complex types.

- Simple types: these include primitive types, arrays composed of elements of primitive types and string types, which can be seeded to construct parameters.

- Interaction types: these are used for functions such as contextual interaction and include the following seven types: android.app.Application, android.content.Context, java.lang.Object, android.content.Intent, android.app.PendingIntent, android.os.Bundle and android.app.Activity.
- Complex types: these include types defined by the software developer and types from third party libraries, such as Map.

In part B, on average, there are 4.8 parameters per native method in malware. Of these, 57.9% of the parameter types are simple types, 40.7% are interactive types (40.36% of which is java.lang.Object type), and 1.3% are complex types. On average, each native method in benign app has 3.1 parameters. 82.3% of them are simple types, 1.2% are interactive types, and 16.5% are complex types. Overall, the overwhelming majority (i.e., 98.6%) of native method parameter types in malware are simple and interactive. In contrast, 16.5% of native functions parameter types are complex type in benign.

Part C indicate that, in malicious apps, there are 62% of so files with static registration functions (export symbol contains: the prefix Java_) and 47% of so files with dynamic registration functions (export symbol contains: JNI_OnLoad), while in benign apps, they are 20.1% and 25.8%, respectively. This means that the benign apps are tend to reuse third-party libraries.