

Method, Storage Medium and Device for Avoiding Fixed Resources in Android OS

TECHNICAL FIELD

The present invention relates to the field of mobile communication technologies, and in particular, to a method, storage medium and device for avoiding fixed resources in an Android OS.

BACKGROUND OF THE INVENTION

With development of terminal devices and computer technologies, software applications based on Android OS are gradually increasing. Google officially provides an official version of resource management scheme to meet needs of developers to manage use of resources. On this basis, the developers can, based on their own product requirements, customize personalized design resources to present to users. However, because resources in an Android OS are arranged alphabetically, resources newly added by a developer can be inserted into a sequence of official resources, resulting in a change in an identification (ID) of the official resources at an insertion position. In addition, changes in resources also affect changes in other ordered resources, resulting in incorrect resource invocation and resource mismatching, and in serious cases, programs and the system cannot operate properly.

To resolve the foregoing problems, the industry has implemented a resource fixing scheme, so as to ensure matching of resources and normal operation of programs and systems. However, the resource fixing scheme has the following disadvantages:

1. Because fixed values of resources cannot be repeated, resource conflicts are likely to occur when one host corresponds to a plurality of patches;
2. When a plurality of sources that need fixed resources are required, the sources need to be fixed uniformly, which is very expensive; and
3. If a patch is compatible with a plurality of hosts, compromise is involved, which increases access costs and reduces compatibility.

SUMMARY OF THE INVENTION

To overcome deficiencies in the conventional technology, the present invention aims to provide a method, storage medium and device for avoiding fixed resources in an Android OS, which can resolve the problems of resource confusion, problems caused by maintenance costs and access costs, and problems of compatibility and compromise of multiple patches.

To achieve the foregoing objective, the present invention uses the following technical solutions:

A method for avoiding fixed resources in an Android OS, including

three stages:

I. when compiling, processing resources of a host referenced in java code, obtaining R.jar and modifying bytecode thereof, removing all final fields from class, and modifying an inline value in the class to ensure that all resource IDs of a host referenced in a plug-in are referenced by using R;

II. at run time, processing resources of a host referenced in xml code, obtaining ap_file by using aapt2, finding corresponding target information according to R.txt, and packaging into apk; and

III. when installing, modifying xml and resources.arsc files in a patch based on a mapRes method according to a compiled configuration file, and modifying a byte array of apk.

Further, the Stage I is specifically implemented as follows:

1.1. writing gradle plugin, obtaining an R.jar file from doLast of processResourcesTask, and obtaining references to host resources in java;

1.2. modifying R.jar bytecode;

1.3. removing all final modifiers of fields whose IDs start with 0x7f in the field, so as to ensure that all targets referencing host resources in the class of the patch are not inlined into numerical values; and

1.4. referencing target resource IDs by using R, writing transform, scanning locations of resources referenced by using R in the patch, changing an original R reference into method invocation by using asm, and invoking a method instead of obtaining a constant value at run time to

internally and dynamically calculate corresponding target resource IDs in a current host.

Further, the Stage II is specifically implemented as follows:

2.1. obtaining references to host resources in xml;

2.2. obtaining ap_file by using aapt2;

2.3. finding a corresponding target resource name according to an R.txt file generated from the aapt2, and recording the resource name, an ID value and a file relationship in a file; and

2.4. packaging the file obtained in step 2.3 into apk.

Further, the Stage III is specifically implemented as follows:

3.1. when installing, obtaining a configuration file generated during compilation; and

3.2. modifying a bbyte array for xml and resources.arsc files based on a mapRes method, reading/writing a file with a RandomAccessFile tool provided by java, decompressing the file by using a Deflate algorithm, modifying file metadata, and then compressing; and avoiding changing a file offset, and controlling the file not to be compressed during packaging.

The present invention further provides a computer-readable storage medium, the computer-readable storage medium stores a computer program, and the computer program, when executed by a processor, performs the foregoing method.

The present invention further provides a computer device, including a

processor and a memory, the memory is configured to store a computer program, and the processor is configured to perform the foregoing method when the computer program is executed.

The present invention has the following beneficial effects: By using the method provided in the present invention, a patch can always be consistent with resources of a host without constraining the host; and a resource ID of a target file in the current host can be dynamically obtained by mapping an established reference into a target string and then using a Resource#getIdentifier method provided by an Android OS, so as to ensure that the patch is loaded to a correct resource. Therefore, the problems of mismatching and instability of resource references caused by other factors are resolved, and dynamic matching ensures stability of target references, reduces access costs and resolves the problems of compatibility and compromise caused by fixed resources.

BRIEF DESCRIPTION OF DRAWINGS

FIG. 1 is a schematic flow chart of a method according to an embodiment of the present invention.

DESCRIPTION OF EMBODIMENTS

The present invention will be further described with reference to accompanying drawings. It should be noted that this embodiment

provides, based on this technical solution, a detailed implementation and a specific operation process, but the protection scope of the present invention is not limited to this embodiment.

This embodiment provides a method for avoiding fixed resources in an Android OS. As shown in FIG. 1, the method includes three stages:

I. when compiling, processing resources of a host referenced in java code, obtaining R.jar and modifying bytecode thereof, removing all final fields from class, and modifying an inline value in the class to ensure that all resource IDs of a host referenced in a plug-in are referenced by using R;

II. at run time, processing resources of a host referenced in xml code, obtaining ap_file by using aapt2, finding corresponding target information according to R.txt, and packaging into apk; and

III. when installing, modifying xml and resources.arsc files in a patch based on a mapRes method according to a compiled configuration file, and modifying a byte array of apk.

Specifically, the Stage I is specifically implemented as follows:

1.1. writing gradle plugin, obtaining an R.jar file from doLast of processResourcesTask, and obtaining references to host resources in java;

1.2. modifying R.jar bytecode;

1.3. removing all final modifiers of fields whose IDs start with 0x7f in the field, so as to ensure that all targets referencing host resources in the class of the patch are not inlined into numerical values; and

1.4. referencing target resource IDs by using R, writing transform, scanning locations of resources referenced by using R in the patch, changing an original R reference into method invocation by using asm, and invoking a method instead of obtaining a constant value at run time to internally and dynamically calculate corresponding target resource IDs in a current host.

Specifically, the Stage II is specifically implemented as follows:

2.1. obtaining references to host resources in xml;

2.2. obtaining ap_file by using aapt2;

2.3. finding a corresponding target resource name according to an R.txt file generated from the aapt2, and recording the resource name, an ID value and a file relationship in a file; and

2.4. packaging the file obtained in step 2.3 into apk.

Specifically, the Stage III is specifically implemented as follows:

3.1. when installing, obtaining a configuration file generated during compilation; and

3.2. modifying a byte array for xml and resources.arsc files based on a mapRes method, reading/writing a file with a RandomAccessFile tool provided by java, decompressing the file by using a Deflate algorithm, modifying file metadata, and then compressing; and avoiding changing a file offset, and controlling the file not to be compressed during packaging.

In this case, an increase in a package volume caused by uncompression and

modified references can be ignored, because the patch is generally not too large.

By using the foregoing method, an ID referenced to a host resource in a final patch is processed in three stages, and a reference is converted into a target ID resource, thereby resolving the problems of mismatching and instability of resource references caused by other factors, and dynamic matching ensures stability of target references, reduces access costs and resolves the problems of compatibility and compromise caused by fixed resources.

For a person skilled in the art, corresponding changes and variants can be given according to the foregoing technical solutions and concepts, and all these changes and variants should be included in the protection scope of the claims of the present invention.

CLAIMS

1. A method for avoiding fixed resources in an Android OS, comprising three stages:

I. when compiling, processing resources of a host referenced in java code, obtaining R.jar and modifying bytecode thereof, removing all final fields from class, and modifying an inline value in the class to ensure that all resource IDs of a host referenced in a plug-in are referenced by using R;

II. at run time, processing resources of a host referenced in xml code, obtaining ap_file by using aapt2, finding corresponding target information according to R.txt, and packaging into apk; and

III. when installing, modifying xml and resources.arsc files in a patch based on a mapRes method according to a compiled configuration file, and modifying a byte array of apk.

2. The method for avoiding fixed resources in an Android OS according to Claim 1, wherein the Stage I is specifically implemented as follows:

1.1. writing gradle plugin, obtaining an R.jar file from doLast of processResourcesTask, and obtaining references to host resources in java;

1.2. modifying R.jar bytecode;

1.3. removing all final modifiers of fields whose IDs start with 0x7f in the field, so as to ensure that all targets referencing host resources in the class of the patch are not inlined into numerical values; and

1.4. referencing target resource IDs by using R, writing transform, scanning locations of resources referenced by using R in the patch, changing an original R reference into method invocation by using asm, and invoking a method instead of obtaining a constant value at run time to internally and dynamically calculate corresponding target resource IDs in a current host.

3. The method for avoiding fixed resources in an Android OS according to Claim 1, wherein the Stage II is specifically implemented as follows:

2.1. obtaining references to host resources in xml;

2.2. obtaining ap_file by using aapt2;

2.3. finding a corresponding target resource name according to an R.txt file generated from the aapt2, and recording the resource name, an ID value and a file relationship in a file; and

2.4. packaging the file obtained in step 2.3 into apk.

4. The method for avoiding fixed resources in an Android OS according to Claim 1, wherein the Stage III is specifically implemented as follows:

3.1. when installing, obtaining a configuration file generated during compilation; and

3.2. modifying a bbyte array for xml and resources.arsc files based on a mapRes method, reading/writing a file with a RandomAccessFile tool

provided by java, decompressing the file by using a Deflate algorithm, modifying file metadata, and then compressing; and avoiding changing a file offset, and controlling the file not to be compressed during packaging.

5. A computer-readable storage medium, wherein the computer-readable storage medium stores a computer program, and the computer program, when executed by a processor, performs the method according to any one of Claims 1 to 4.

6. A computer device, comprising a processor and a memory, wherein the memory is configured to store a computer program, and the processor is configured to perform the method according to any one of Claims 1 to 4 when the computer program is executed.