black hat ASIA 2015

We can still crack you!

General unpacking method for Android Packer(NO ROOT)

black hat ASIA 2015

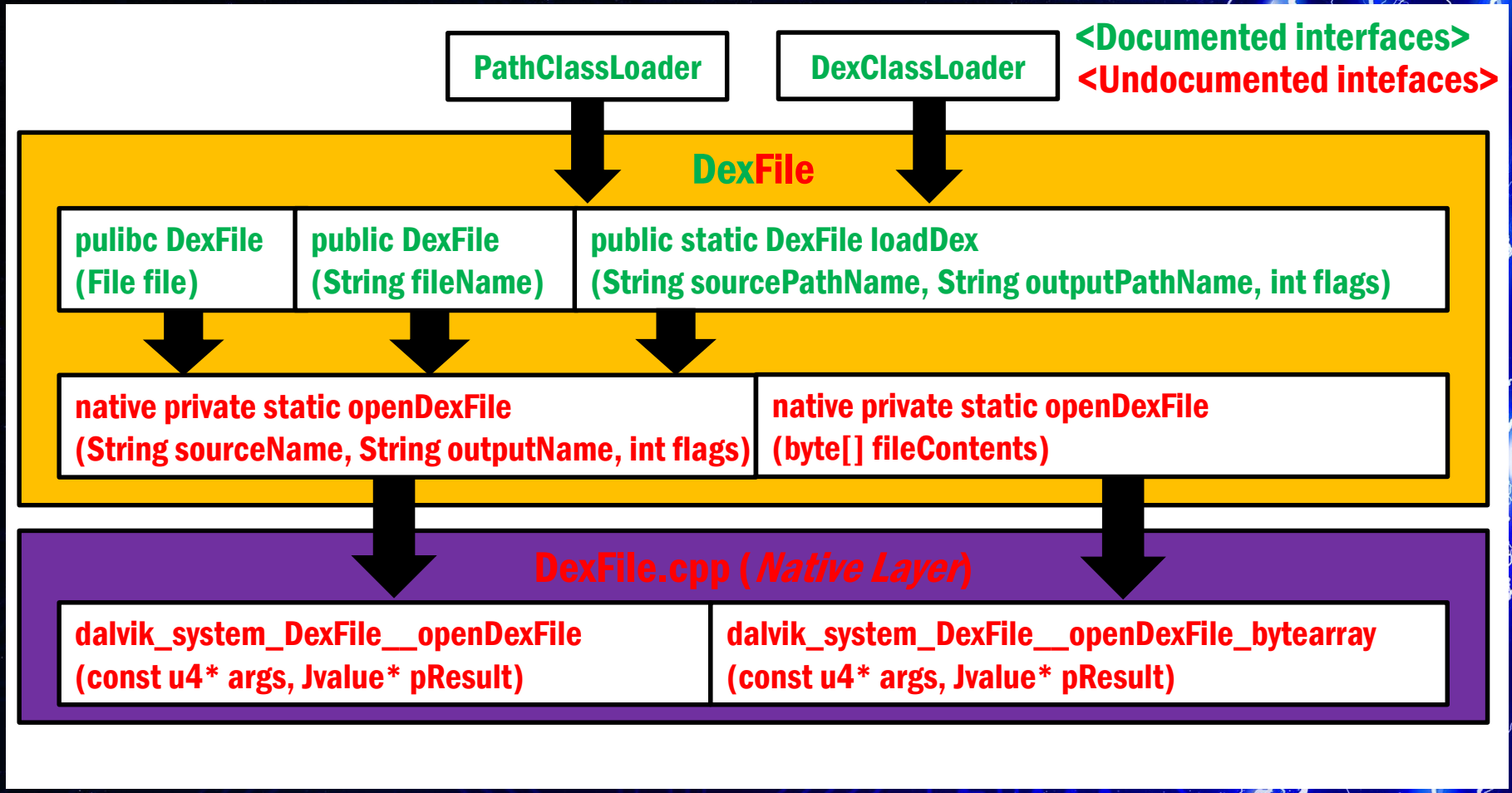
What is android packer?

- Android packer is similar to UPX
- There are several commercial android packers (Ijiami, BangCle, DexGuard, LIAPP, etc)
- They are distinguished two types by main packing mechanism
  - *Dynamic code(*.dex/jar/apk) loading* based
  - *Memory patch* based
- There are various papers for features of android packers

Packing mechanism

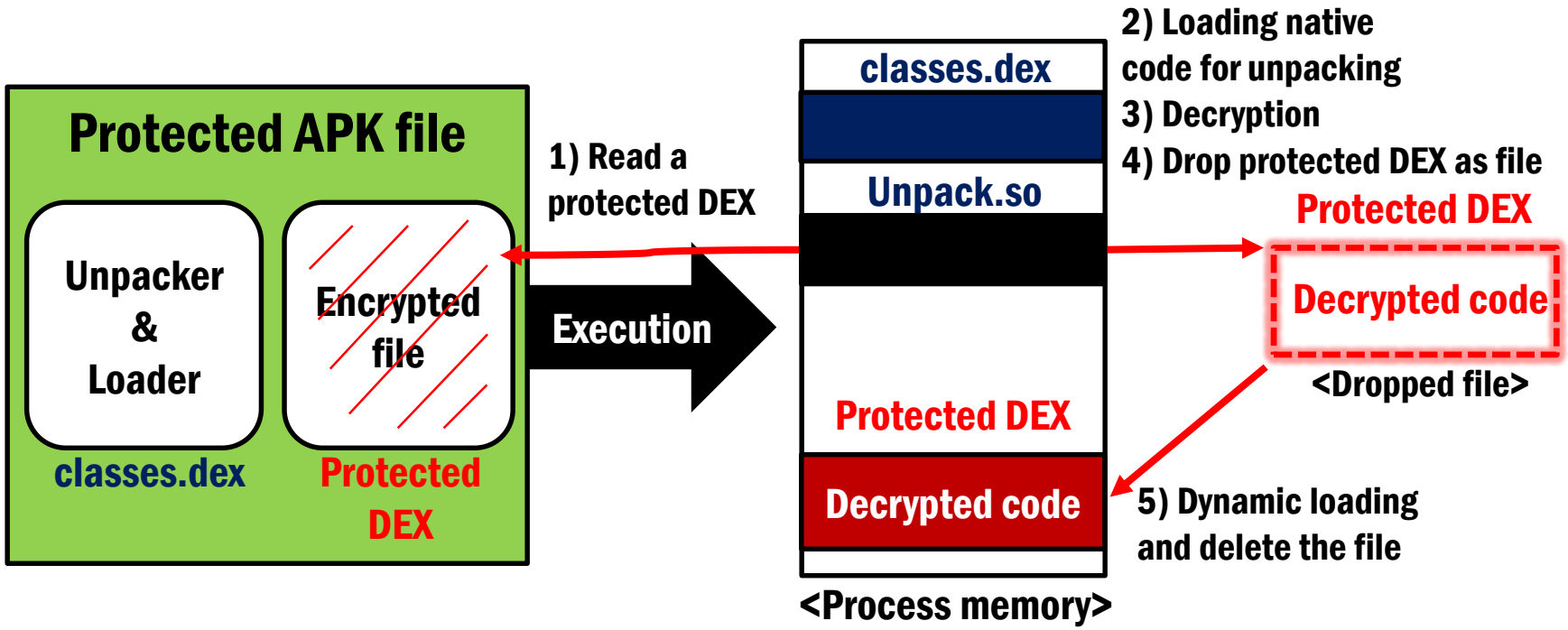
- Packing mechanism based on Dynamic code loading
  - It can load code ***in file*** *or* ***on memory*** dynamically
  - Android platform provides following interfaces only for Java layer to load .dex file dynamically
    - Documented interfaces: *DexClassLoader, PathClassLoader, DexFile*
    - Undocumented interfaces:
      - ***DexFile.java:***
        - openDexFile(byte[] fileContents)
        - openDexFile(String sourceName, String outputName, int flags)
      - ***dalvik_system_DexFile.cpp:***
        - Dalvik_dalvik_system_DexFile_openDexFile(const u4* args, Jvalue* pResult)
        - Dalvik_dalvik_system_DexFile_openDexFile_bytearray

Packing mechanism



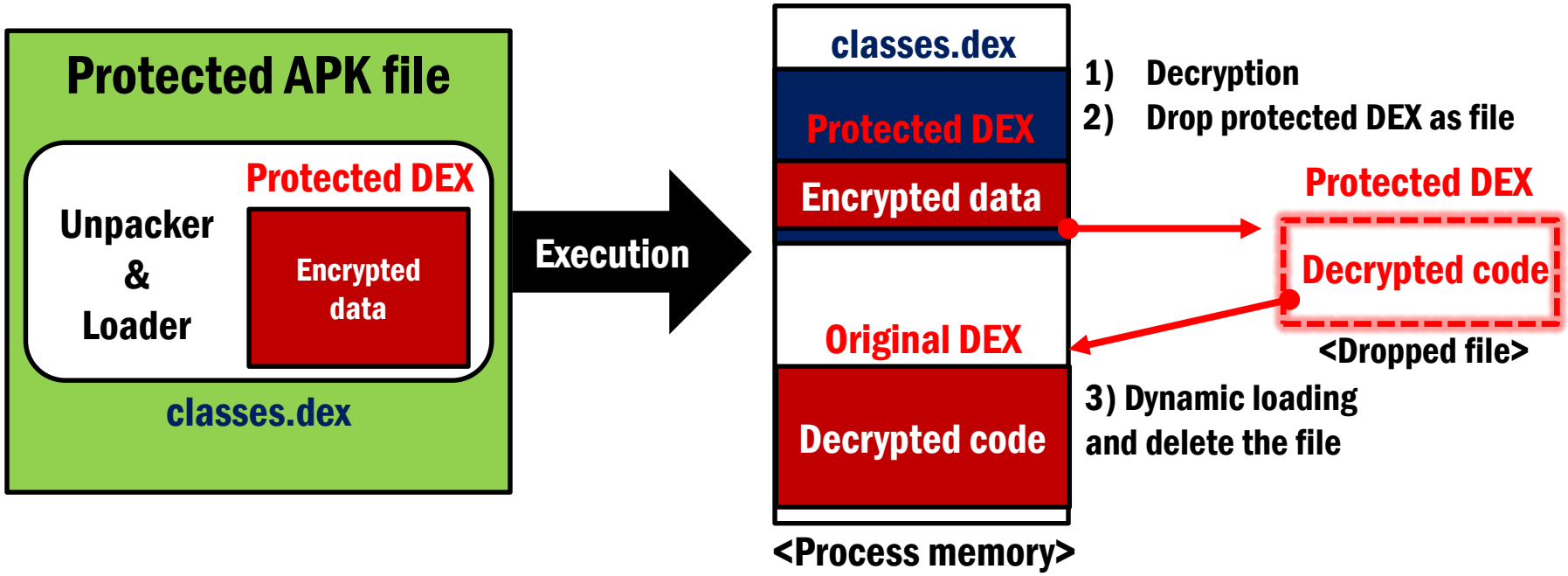
Packing mechanism

- Dynamic code loading (***in file***)
  - Many android packer are using this method



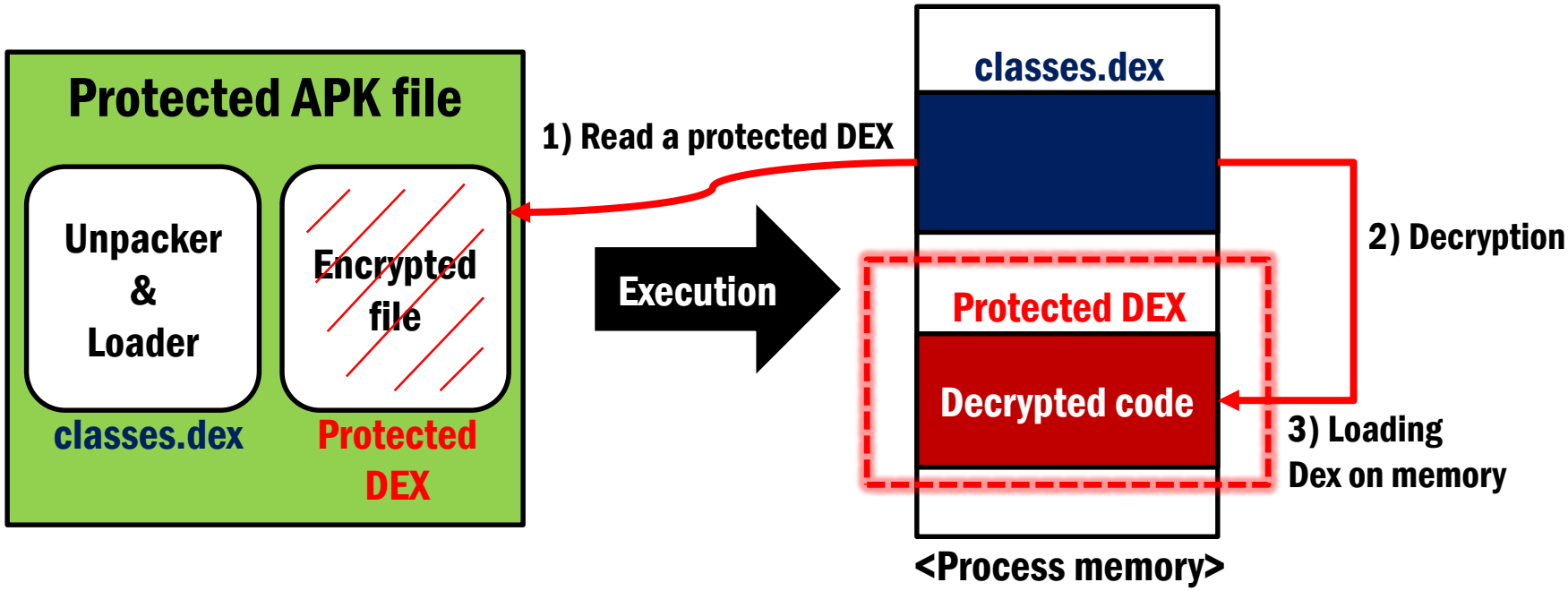
Packing mechanism

- Dynamic code loading (***in file***)
  - Protected DEX is in unpacking dex file as array



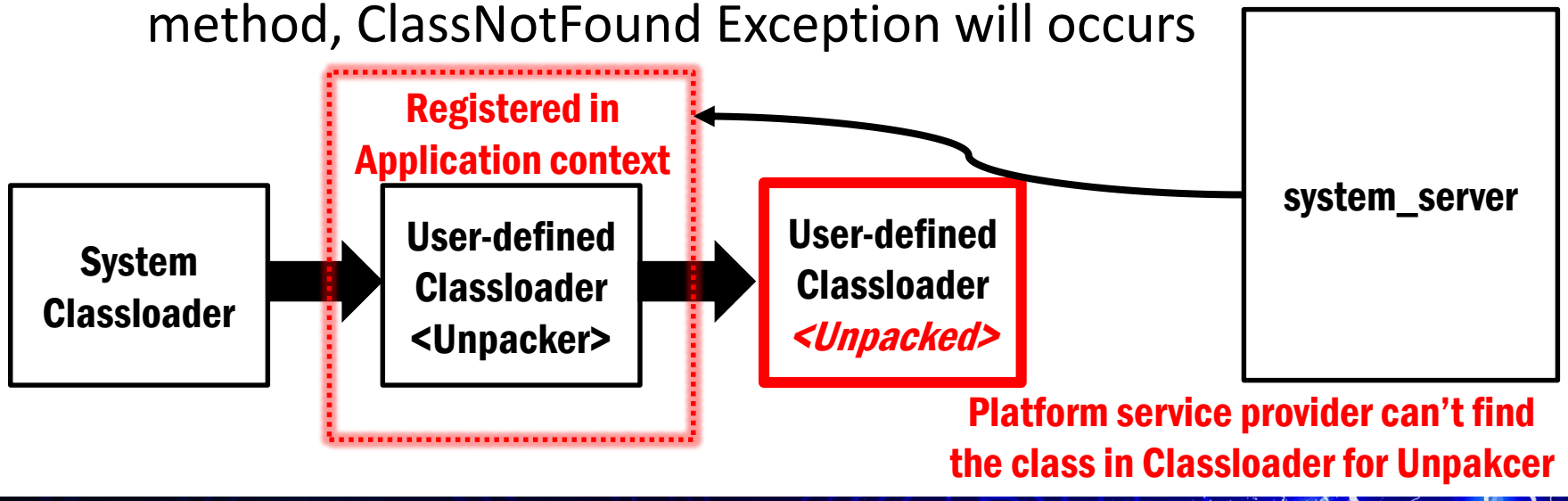
Packing mechanism

- Dynamic code loading (***on memory***)



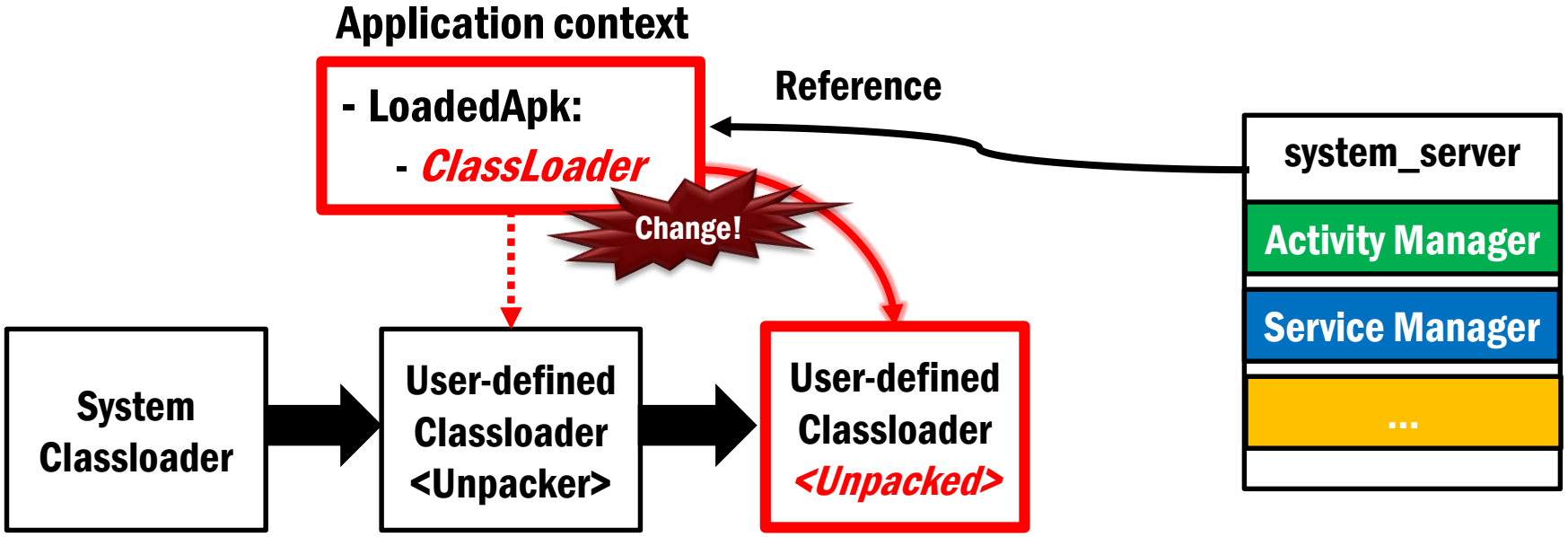
Packing mechanism

- Loading a separated DEX file dynamically causes ClassLoader problem
  - When decrypted dex loaded by different class loader from class loader in Application context try to load and call some method, ClassNotFound Exception will occurs



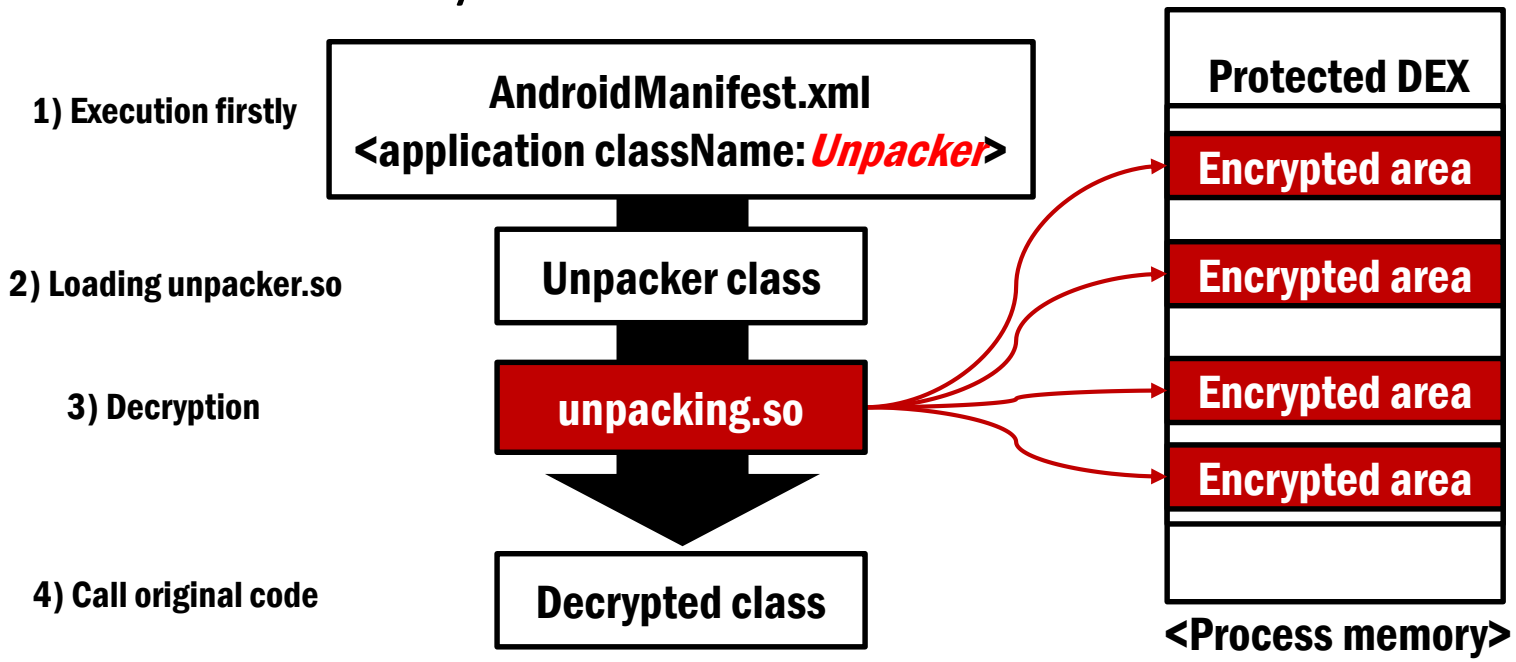
Packing mechanism

- Context is a key for execution of main components in Android application (Activity, Service, Receiver, etc)
  - Unpacker needs to change a object of ClassLoader in Application context to execute unpacked code correctly



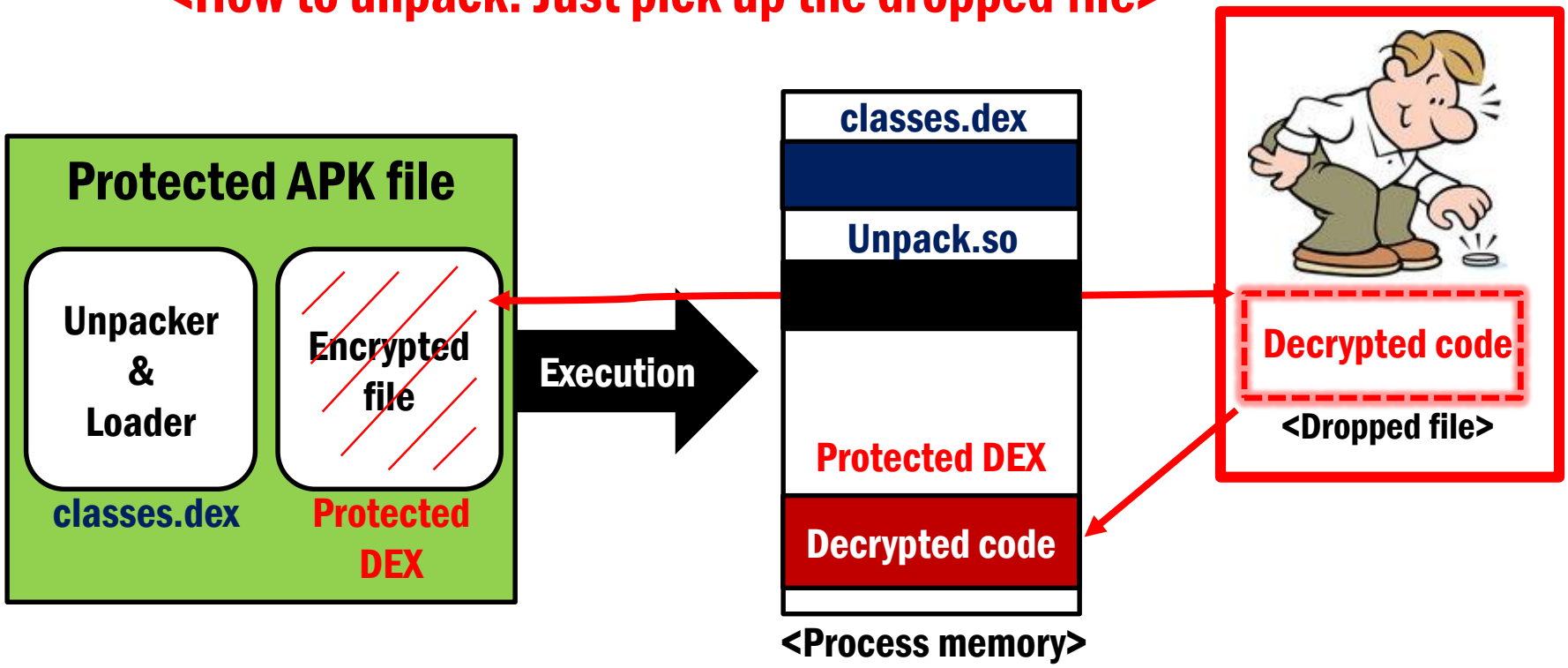
Packing mechanism

- Packing mechanism based on Memory patch
  - It modifies <application> tag in AndroidManifest.xml to be executed firstly



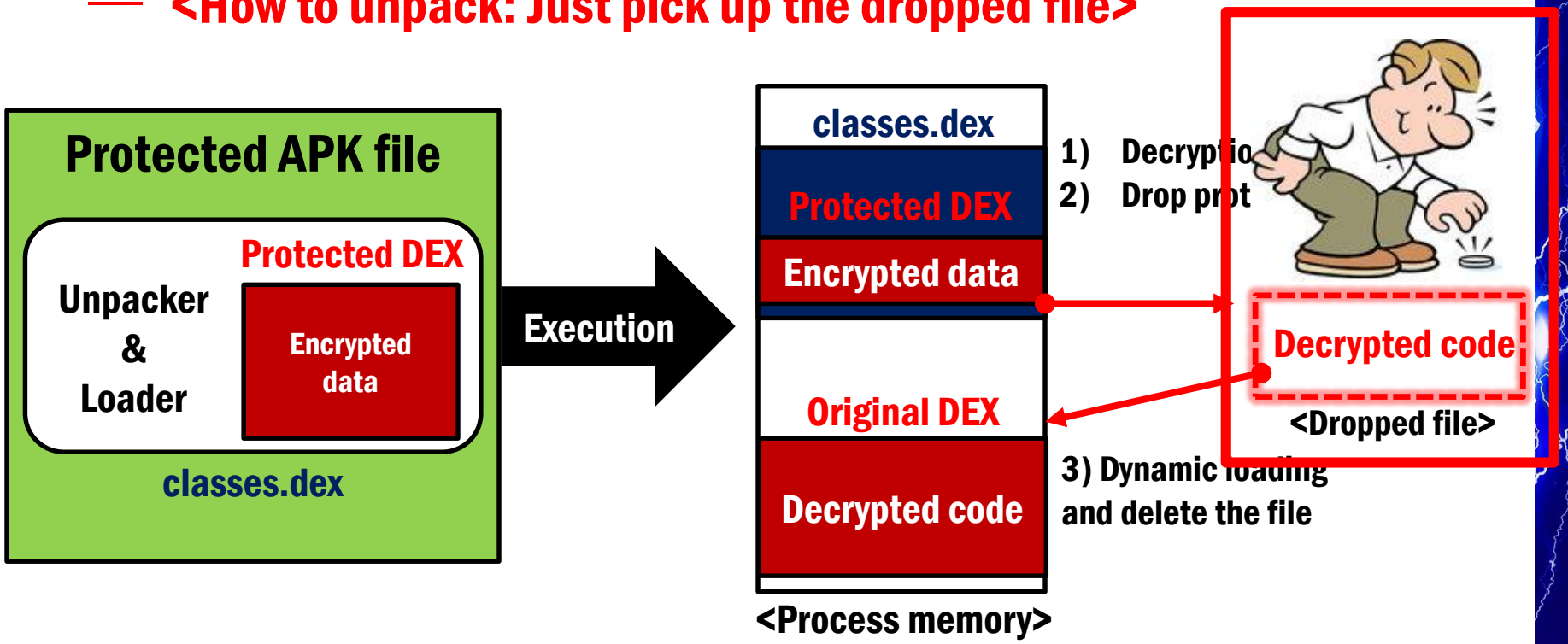
How to unpack logically

- Dynamic code loading (***in file***)
  - **<How to unpack: Just pick up the dropped file>**



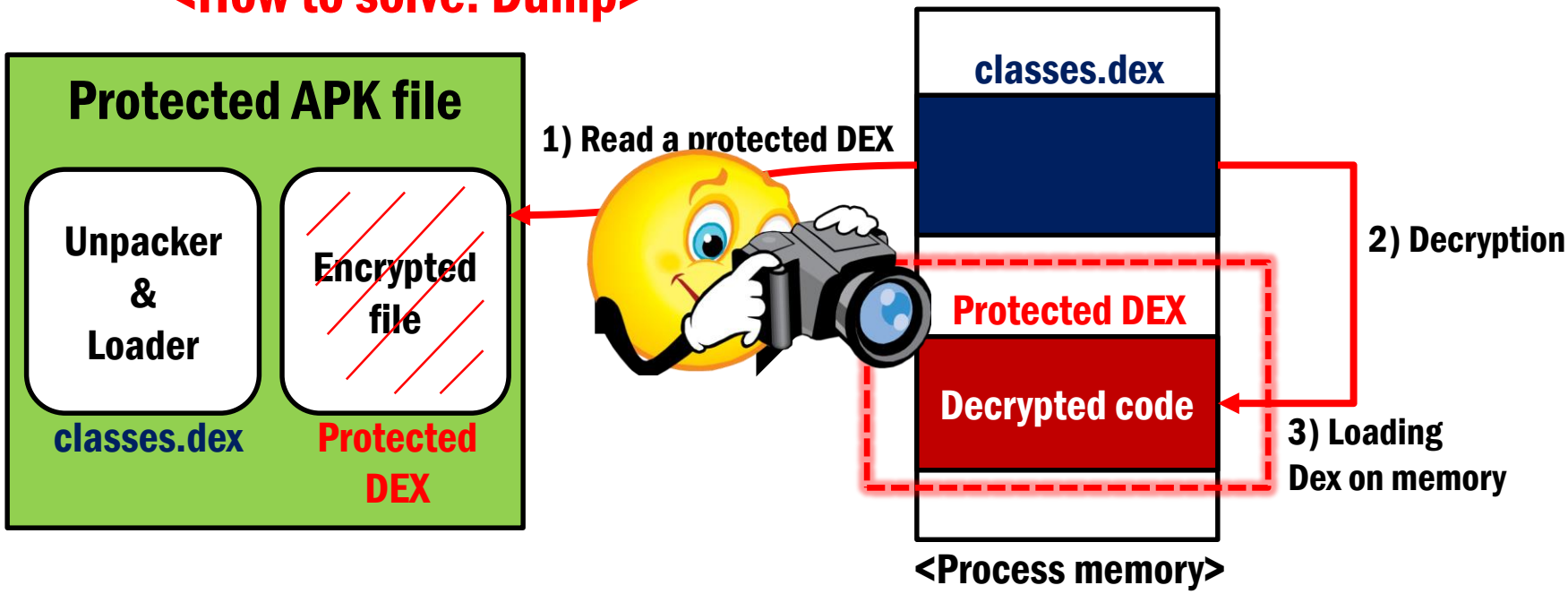
How to unpack logically

- Dynamic code loading (***in file***)
  - **<How to unpack: Just pick up the dropped file>**



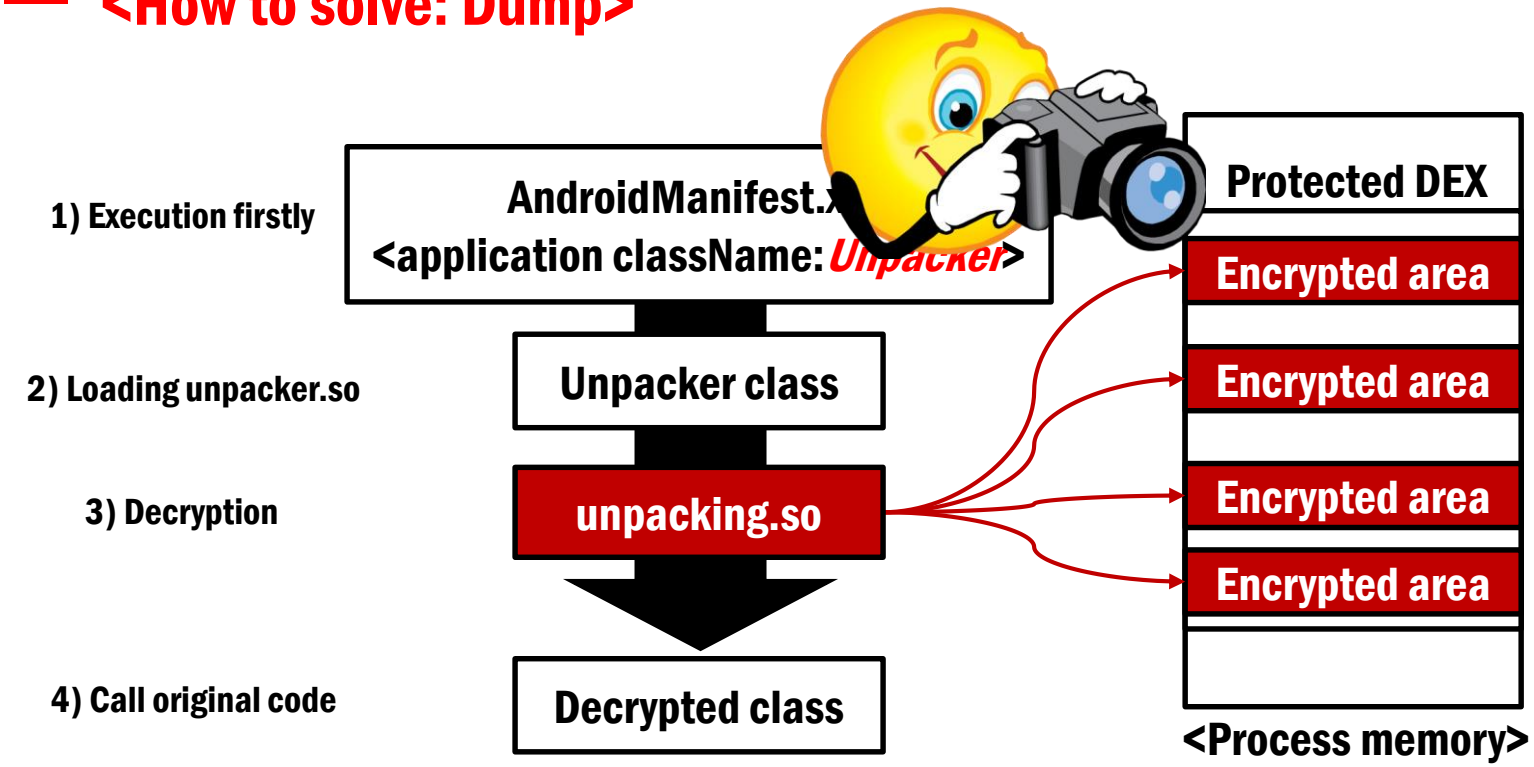
How to unpack logically

- Dynamic code loading (***on memory***)
  - **<How to solve: Dump>**



How to unpack logically

- Packing mechanism based on Memory patch
  - **<How to solve: Dump>**



Challenges to unpack

- Anti-debugging (for gdb, ptrace)
- Anti-debugging (for JDWP)
- Emulator/Device detection
- Rooting detection
- Obfuscation
- Native-level behavior
- Self integrity check


black hat
ASIA 2015
Now
Let's unpack ☺
black hat
ASIA 2015

How to unpack practically

- Each challenge can be overcome
- Real-world packed android application is being applied many challenges multiply
- We can utilize multiple solutions for multiple challenges

How to unpack: Condition

- We have to satisfy following conditions to unpack easily
  - — Don’t use android emulator
  - — Don’t require root privilege

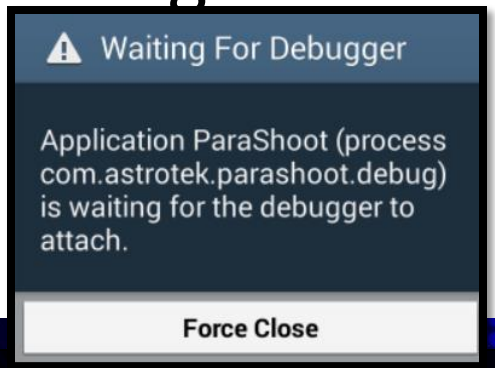
  Use real-device without root, your device.. ☺
  - — Don’t use debugger
  - — Don’t use JDWP

  Yeah.. just don’t use
  - — Don’t analyze obfuscated unpacking stub
  - — Pick up coin and dump
  - — Make your own process environment

  Hooking!

How to unpack: wait-for-debug

- Android platform provides wait-for-debug feature to debug android application
- ActivityManager provides a function makes android application wait for connection for JDWP at starting point using command “wait-for-debug”
- We need to repackage the protected application to use wait-for-debug feature



How to unpack: Process environment to unpack and trace

- When the debuggee is waiting for debugger at starting point of Android application, DEX file is not loaded on memory
- There is MethodEntryEvent in JDWP
- We can control a threads suspended by jdwp event
- We can control the execution of debuggee using wait-for-debug feature and MethodEntryEvent before the DEX file is loaded on memory

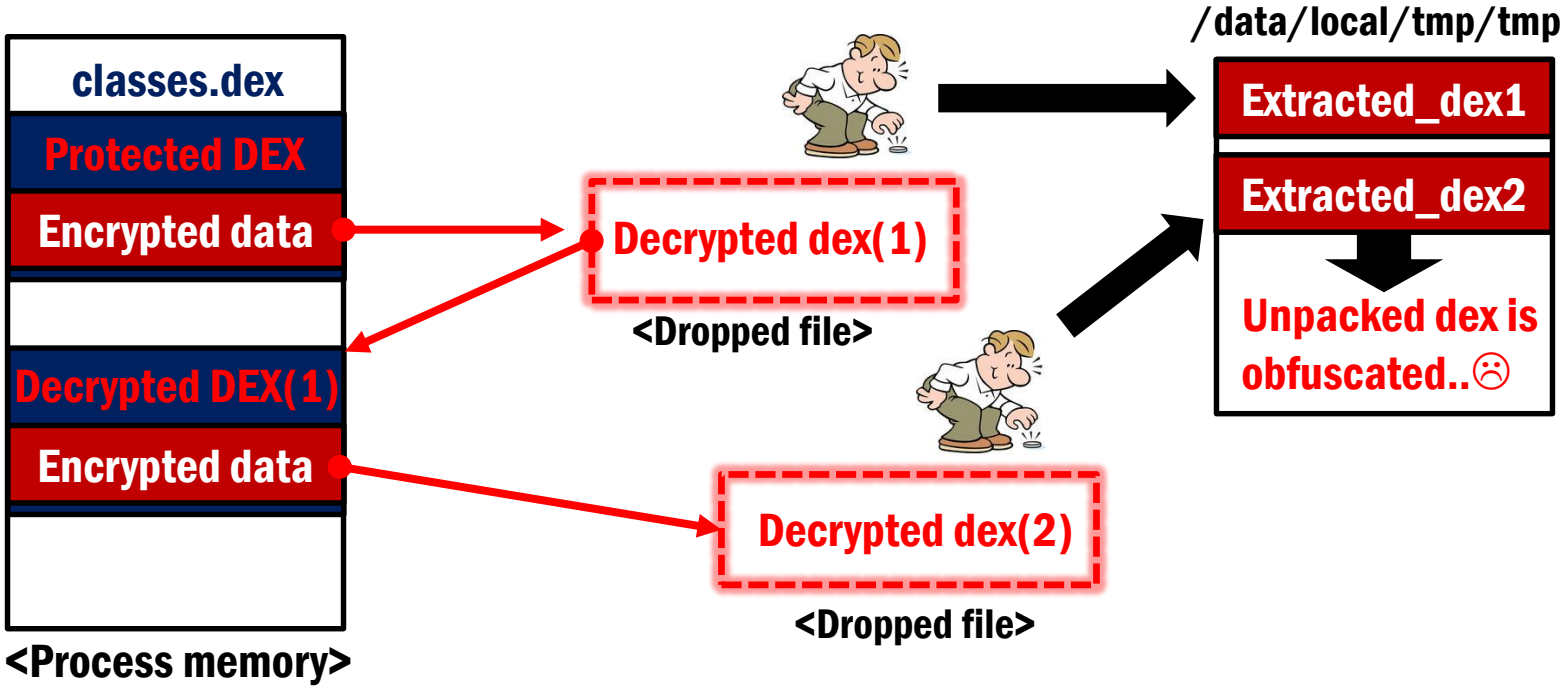
Unpacking: DexGuard

- DexGuard is employing dynamic code loading technique for execution of unpacked code
- It can identified using logcat easily

```
D/dalvikvm(11183): DexOpt: --- BEGIN '.遊샊뿽꽈' (bootstrap=0) ---
D/dalvikvm(11183): DexOpt: --- END '.遊샊뿽꽈' (success) ---
D/dalvikvm(11183): DEX prep '/data/data/com.example/.遊샊뿽꽈': unzip in 0ms, re
write 69ms
D/dalvikvm(11183): DexOpt: --- BEGIN '.遊샊뿽꽈' (bootstrap=0) ---
D/dalvikvm(11183): DexOpt: --- END '.遊샊뿽꽈' (success) ---
D/dalvikvm(11183): DEX prep '/data/data/com.example/.遊샊뿽꽈': unzip in 0ms, re
write 69ms
```

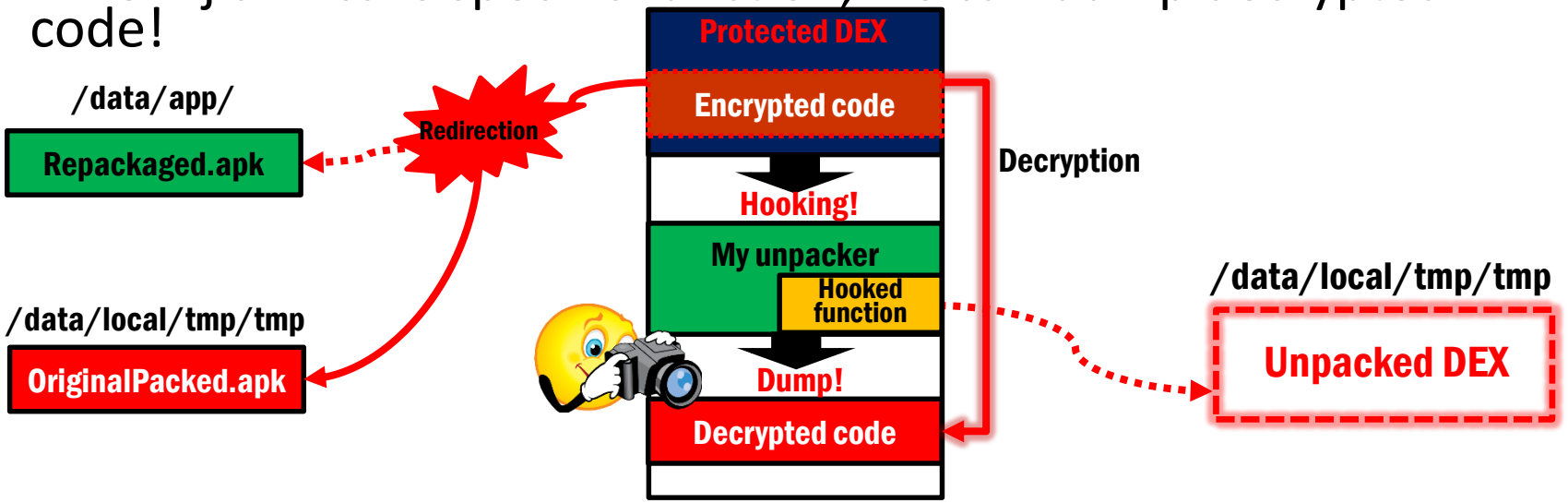
Unpacking: DexGuard

- We can hook various function to pick up coin..☺
- I use hooking open() in libc.so



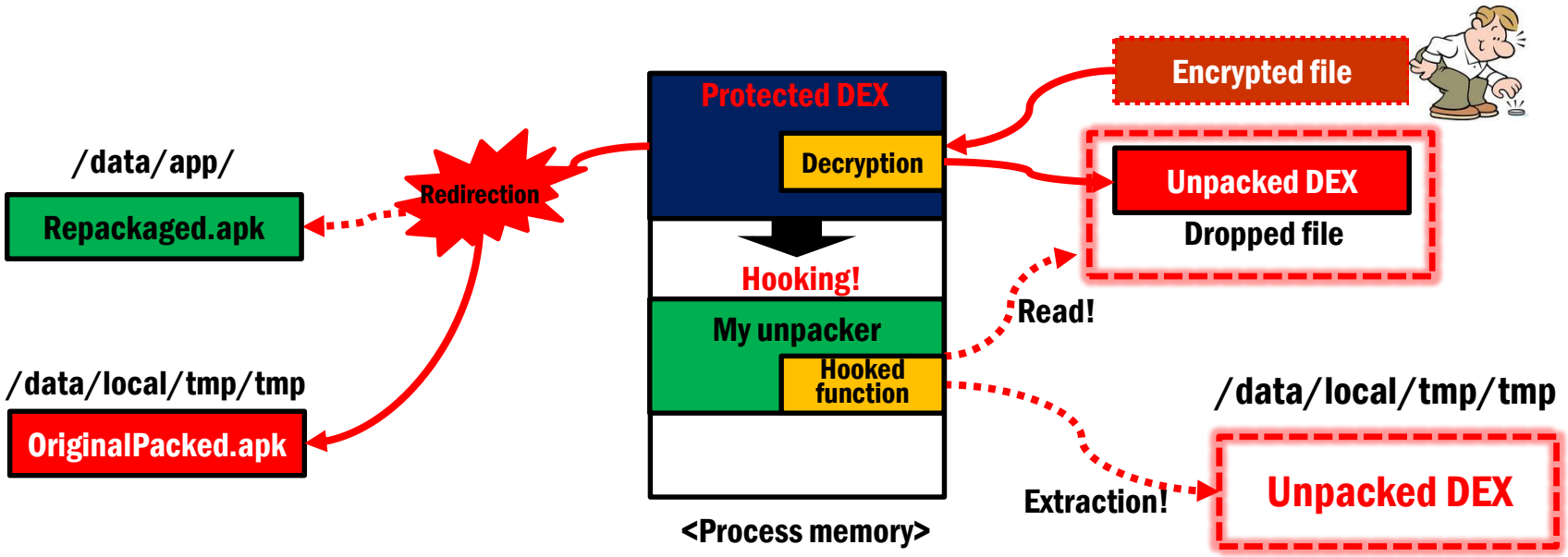
Unpacking: Ijiami

- Ijiami checks integrity of apk file
- I couldn't see dex optimization log with logcat
  - Then, we can dump memory ☺
- When do we need to dump?
  - We can know it by hooking dlopen, dlsym
- When Ijiami calls specific function, we can dump decrypted code!



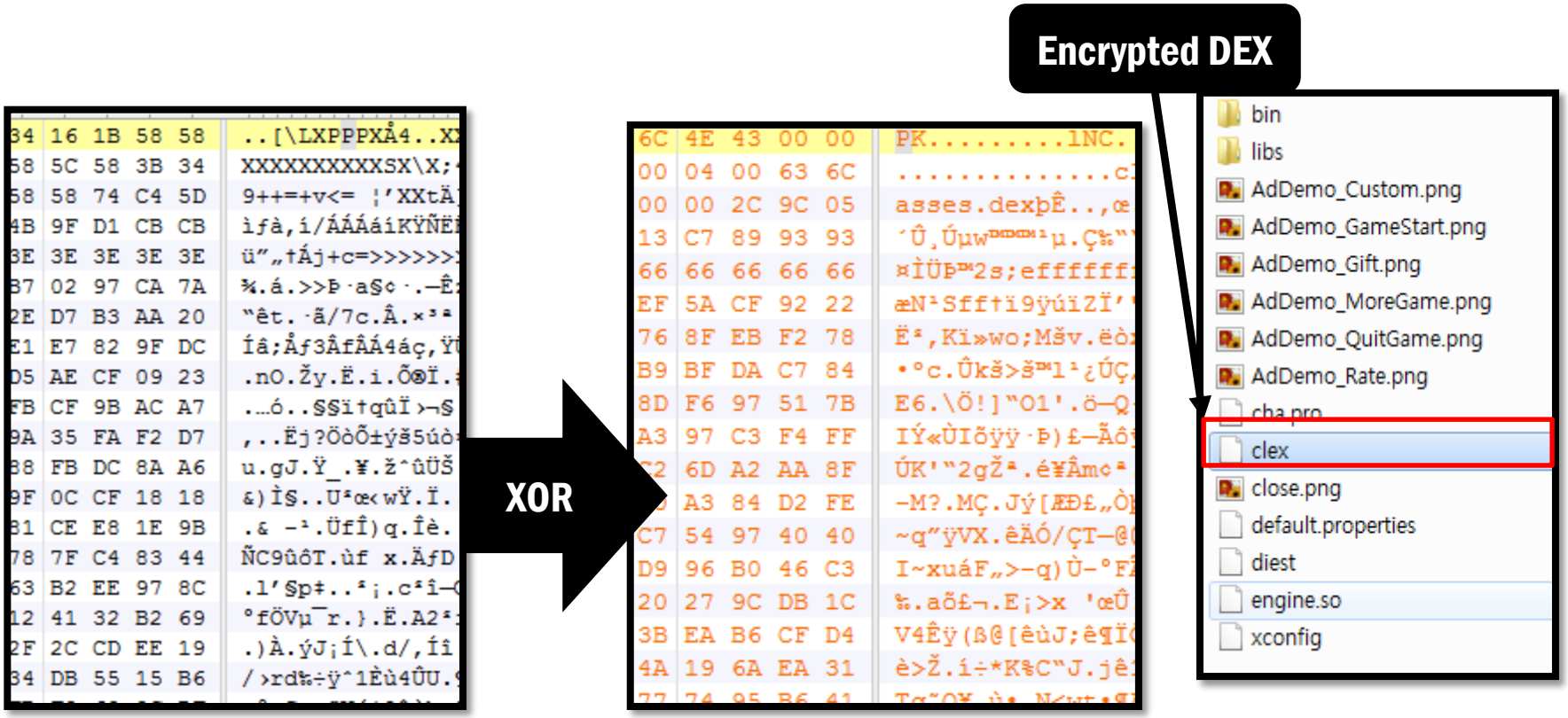
Unpacking: LIAPP

- LIAPP check integrity its .apk file too
- LIAPP uses dynamic code loading
- We can extract unpacked dex



Unpacking: PangXie

- PangXie, just unpack manually..




black hat
ASIA 2015
DEMO:
BangCle
DexProtector
APKProtect
black hat
ASIA 2015

Unpacking: BangCle

- BangCle unpacks encrypted dex file and loads it
- BangCle performs unpacking, anti-analysis and integrity checking simultaneously with multiple threads

Unpacking: DexProtector

- DexProtector is using dynamic code loading
- DexProtector employs multiple unpacking step
- It performs integrity checking using Signature class in PackageInfo

Unpacking: APKProtect

- APKProtect performs memory patch to unpack
- It checks integrity of odex file mapped on memory

Conclusion

- You don’t need reversing unpacker’s code
  - Prediction, Tracing based on hooking..
  - ***Use my powerful tool for analysis of android app☺***
- We can unpack most android packers using wait-for-debug feature and injection
- Companies developing android packer need to response to wait-for-debug feature