FridaDroid: Hooking and scalable patching of mobile applications via Android Runtime hooking

Patrick Lavoisier Wapet (patricklavoisier.wapet@enseeith.fr)

National Higher Polytechnic Institute of Toulouse

IRIT laboratory

SEPIA team

Context

- Dynamic analysis of mobile applications: collection information relating to the application during its operation (“at runtime”).
  - Possible uses: classifications, comparisons ...
- Several tools are available
  - Scientific research (ARTDroid) [1]
  - Open Source Projects (Frida) [2]

[1] ARTDroid: AVirtual-Method Hooking Framework on Android ART Runtime [2]

https://www.frida.re/docs/gadget/

Problems

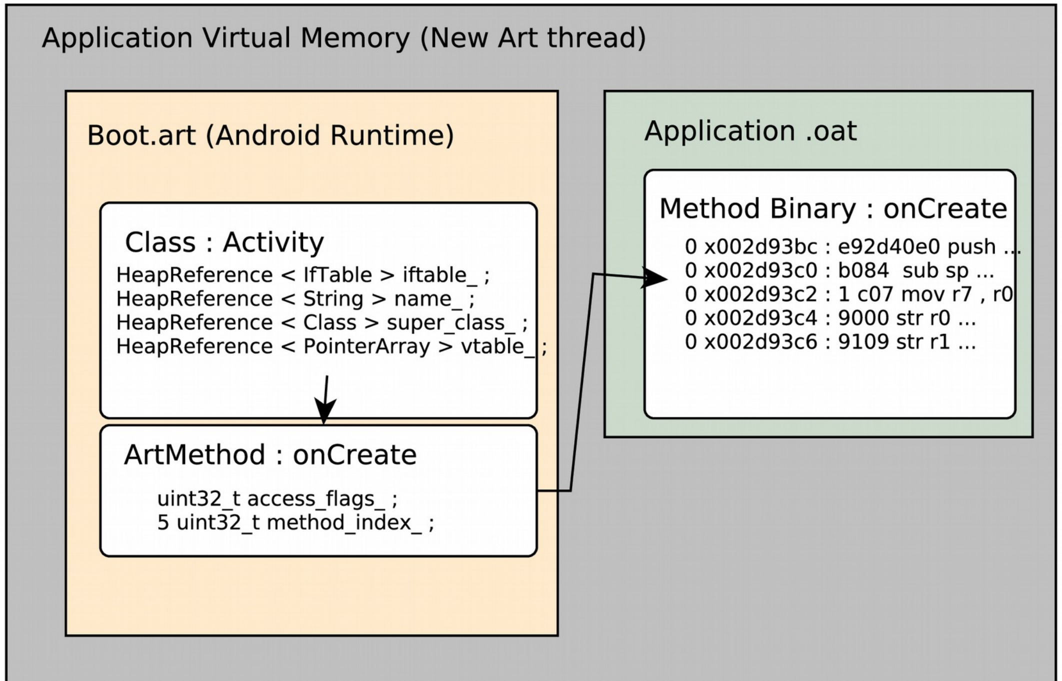
- Problems with Open Source Tools
  - The data is collected on a terminal other than the mobile
  - The collection process is not scalable
- Problem with ARTDroid
  - Need to create a DexFile of the Class we want to analyze. So to know it in its smallest details.
- We offer FridaDroid, a solution based on Frida offering
  - Transparency towards users
  - Scalability
  - Non intrusive to the Android system

Prerequisites - Definitions

- Hoocker (a method): Get a reference / pointer to this method / function, to possibly call or modify it. The reference obtained is called a “hook”.
- Patch (a method): Modify the value of the "hook" obtained at term of hoocking with another implementation called here the “patch”.

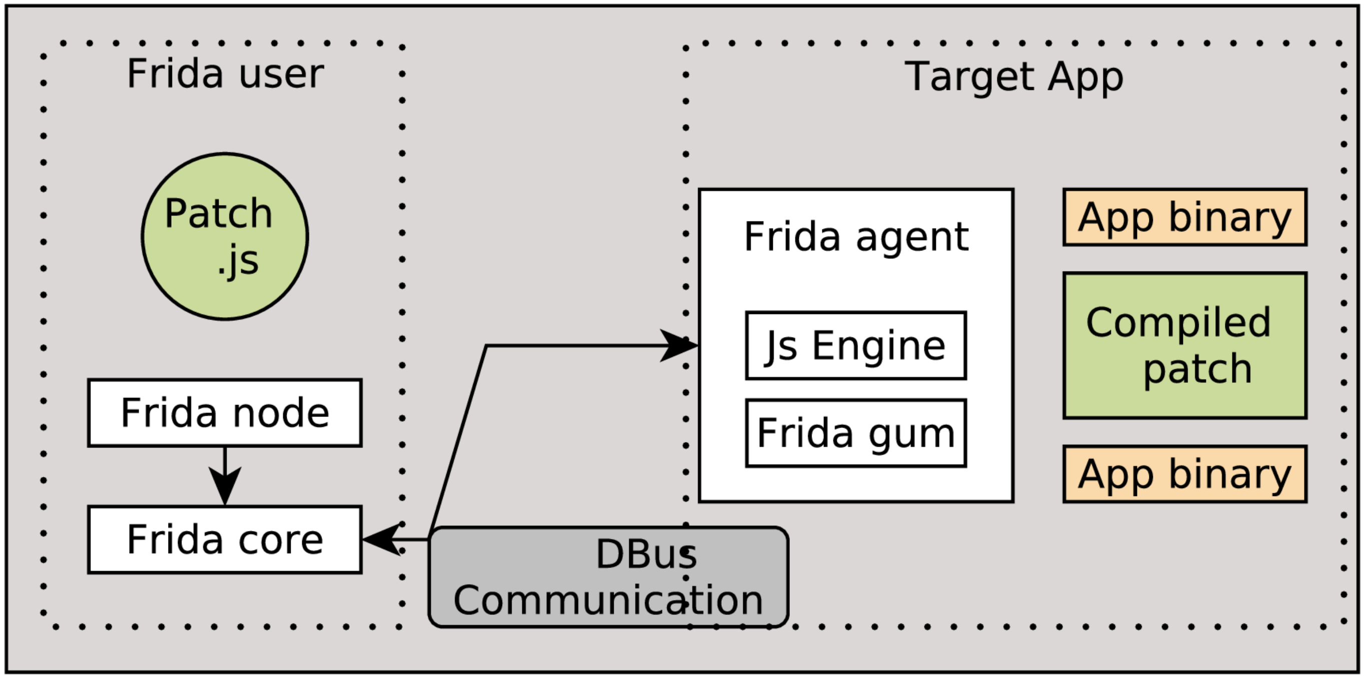
Prerequisites - Android Runtime (ART)

- By default, the ART (Virtual Machine) executes the android applications.
- When running an app, ART keeps track of method implementations.
- For this the ART uses an .oat file which contains the binaries of these methods.


Application Virtual Memory (New Art thread)
Boot.art (Android Runtime)
Class : Activity
HeapReference < IfTable > iftable_ ;
HeapReference < String > name_ ;
HeapReference < Class > super_class_ ;
HeapReference < PointerArray > vtable_ ;
ArtMethod : onCreate
uint32_t access_flags_ ;
5 uint32_t method_index_ ;
Application .oat
Method Binary : onCreate
0 x002d93bc : e92d40e0 push ...
0 x002d93c0 : b084 sub sp ...
0 x002d93c2 : 1 c07 mov r7 , r0
0 x002d93c4 : 9000 str r0 ...
0 x002d93c6 : 9109 str r1 ...

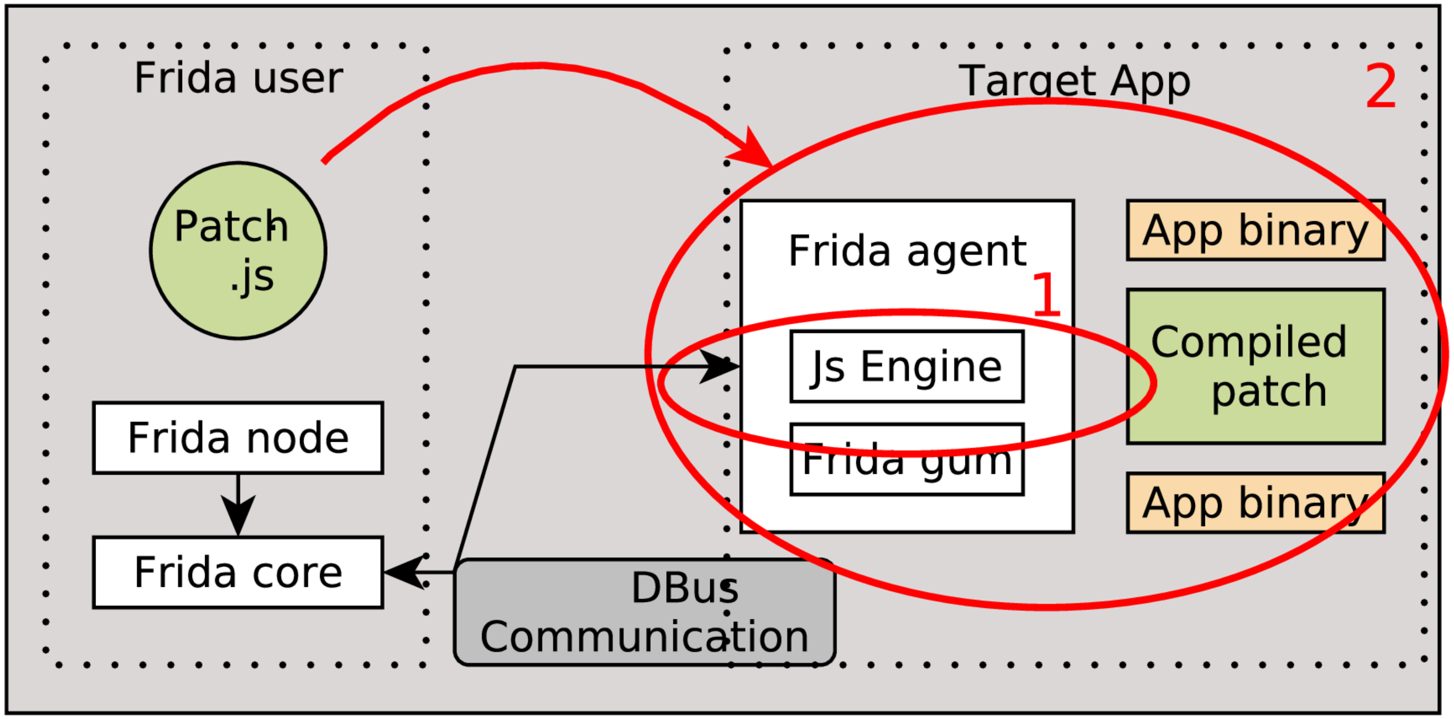
Prerequisite - The Frida tool

- The patched method is sent from the computer of the hacker to the agent frida, the agent being inserted in the application on the mobile.
- And there, the patch (in js) is compiled and used to patch application



Our solution: FridaDroid

- Scalability: We are removing the javascript engine.
- Transparency: The source code of the patch is added to the application before it starts.



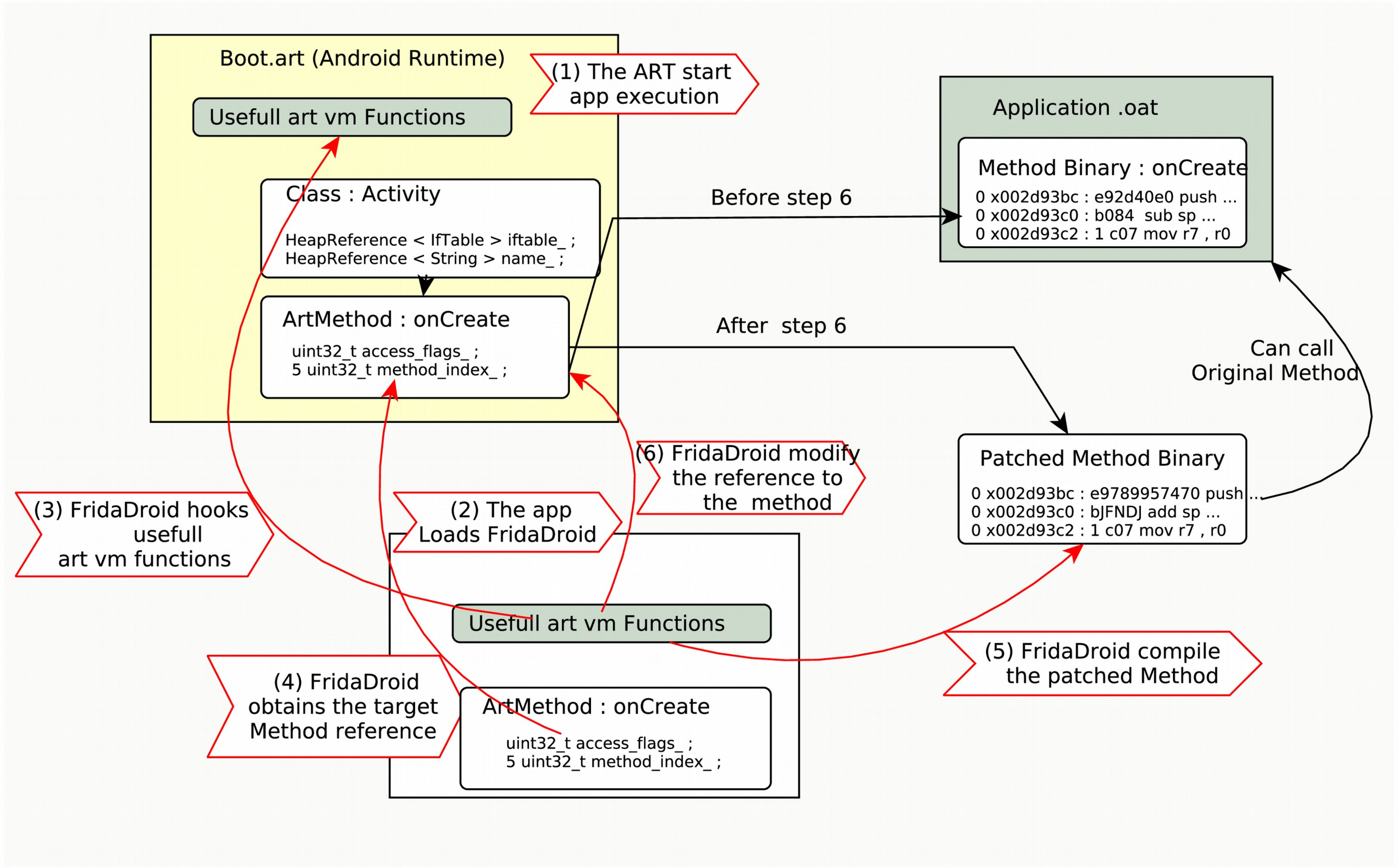
How it works ?

- Before starting the application
  - The source code of the patch is written in C with the glib library and is associated with the application in the form of a library (a .so which contains the APIs of FridaDroid)
  - The app is also instrumented to load this library at startup.

How it works ? (after)

- While the application is running (the user of the phone starts the app)
  - The ART loads the app's .oat file to run it and the .so containing FridaDroid + patch is started.
  - In the .so, FridaDroid
    - (1) hook ART functions
    - (2) Get the references of the target method using the hooks obtained in (1).
    - (3) Compile the patch and modify the reference to the target method with the binaries obtained at the end of the compilation.
  - The application continues to run

Drawing


Boot.art (Android Runtime)
Usefull art vm Functions
Class : Activity
HeapReference < IfTable > iftable_ ;
HeapReference < String > name_ ;
ArtMethod : onCreate
uint32_t access_flags_ ;
5 uint32_t method_index_ ;
(1) The ART start app execution
Application .oat
Method Binary : onCreate
0 x002d93bc : e92d40e0 push ...
0 x002d93c0 : b084 sub sp ...
0 x002d93c2 : 1 c07 mov r7 , r0
Before step 6
After step 6
Can call Original Method
Patched Method Binary
0 x002d93bc : e9789957470 push ...
0 x002d93c0 : bJFNDJ add sp ...
0 x002d93c2 : 1 c07 mov r7 , r0
(6) FridaDroid modify the reference to the method
(3) FridaDroid hooks usefull art vm functions
(2) The app Loads FridaDroid
Usefull art vm Functions
(5) FridaDroid compile the patched Method
(4) FridaDroid obtains the target Method reference
ArtMethod : onCreate
uint32_t access_flags_ ;
5 uint32_t method_index_ ;

How to rate FridaDroid?

- The current objective is the simple collection of information
  - Parameters passed during the call, exceptions, results
- We automatically generate the source codes (using glib) of the method patches.
  - This is done with a python script
  - Script entry: list of signatures of targeted methods (hundreds of signatures !!).
- We instrument the application
- We start it on the android emulator

Results and outlook

- Results
  - Patching of more than 500 functions (currently with fixed signatures).
  - Objective to make patching compatible with all signatures
- Outlook
  - Use the data collected for unsupervised learning (partnership with the University of Rennes)

Thank you for your attention !!!!!