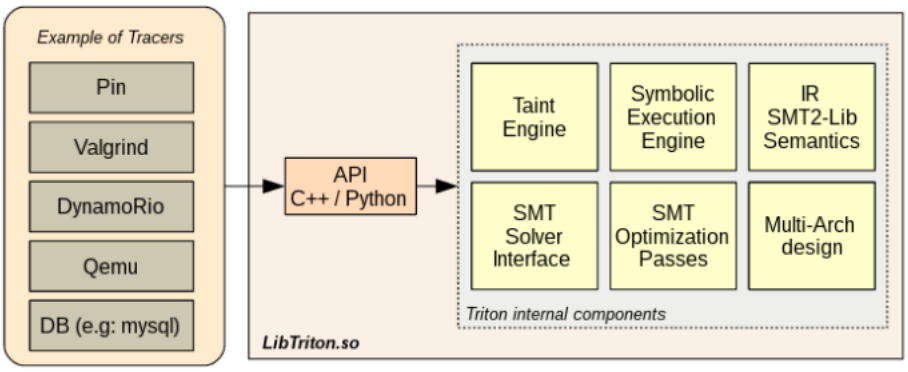
برای بررسی پلتفرم گوشی‌های هوشمند، از برنامه‌های تحت سیستم عامل اندروید استفاده خواهیم کرد. زیرا این سیستم عامل متن‌باز است.

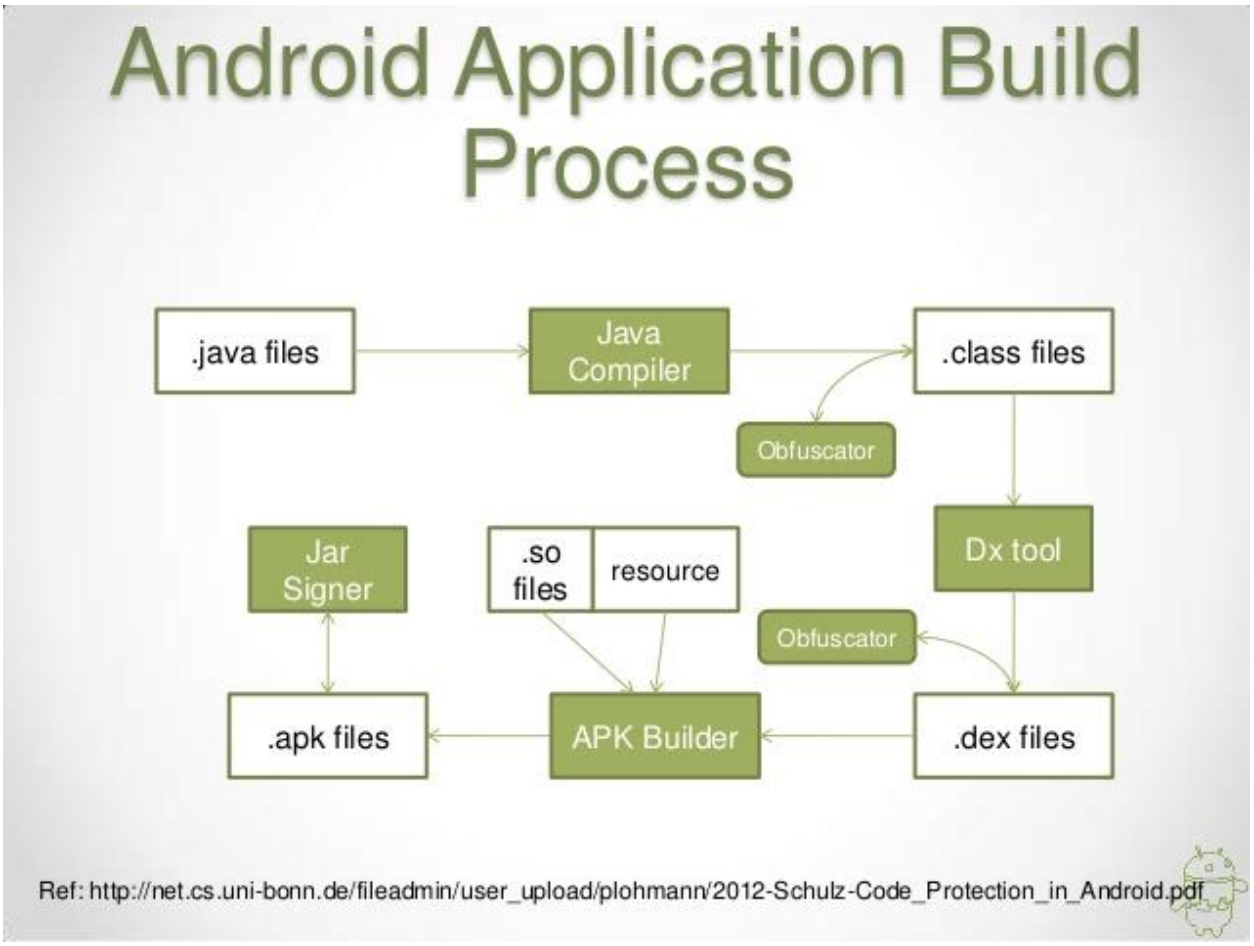
می‌توان از پنج جنبه این برنامه‌ها را تحلیل کرد:

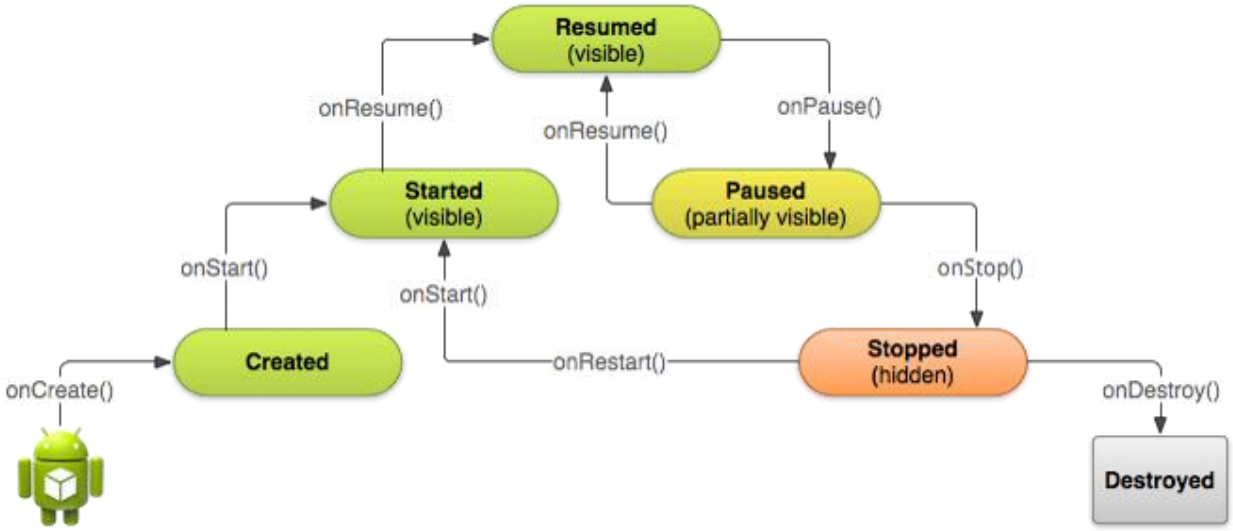
1) بررسی فایل‌های باینری
   - فریم‌ورک‌های Valgrind و Angr از معماری ARM پشتیبانی می‌کنند. ابزاری که در این حوزه هست، CRAXDroid است. این ابزار S2E را توسعه داده است. نکته مهم این است که این ابزار روی Android-x86 پیاده‌سازی شده است. (یک توزیع غیر رسمی از اندروید برای اجرا روی پردازنده‌های x86 هست. پردازنده‌های گوشی‌های هوشمند ARM است. همچنین همه ویژگی‌های ARM در آن وجود ندارد.) ابزار دیگر برای Instrument کردن، REDEXER هست که Dalvik bytecode instrumentation framework است.
2) بررسی برنامه‌های نوشته شده با زبان‌های سطح بالا مثل جاوا
   - نکته مهم این است که حتما نیاز است تا تحلیلی ایستا صورت گیرد تا Call Flow Graph استخراج شود تا بتوان حالت‌های مختلف ورود به برنامه را استخراج کرد.(اندروید بر خلاف جاوا متد main ندارد و حالت‌های مختلفی برای شروع اجرای یک برنامه وجود دارد. مثلا یک رخداد از بیرون. برای تحلیل ایستا هم ابزار soot موجود است.) برای اجرای Concolic هم نیاز است که موتور خاصی وجود داشته باشد. برای این موضوع ابزار Acteve و Condroid (که توسعه همان Acteve برای تحلیل دژافزار هست) به صورت متن‌باز وجود دارند.
3) بررسی برنامه‌های نوشته شده با زبان Native یعنی C.
   - این مورد یعنی آسیب‌پذیری‌های BoF نیز در اندروید امکان دارد اتفاق بیفتند.
4) بررسی برنامه‌های ترکیبی، مجموعه‌ی جاوا و C.
5) بررسی برنامه‌های حاصل از Web Technology، یعنی JS.
   - برای این تحلیل هم نیاز است تا موتور Concolic برای JS وجود داشته باشد که ابزار Jalangi به صورت متن‌باز وجود دارد.

ابزارهای دیگری که در مورد آزمون برنامه‌های اندرویدی وجود دارد و متن‌باز هستند عبارتند از: Monkey، MonkeyRunner و Robotium. این ابزارها برای Fuzz Testing کاربرد دارند.

معماری ابزار Triton:


Example of Tracers
Pin
Valgrind
DynamoRio
Qemu
DB (e.g: mysql)
API
C++ / Python
Taint Engine
Symbolic Execution Engine
IR SMT2-Lib Semantics
SMT Solver Interface
SMT Optimization Passes
Multi-Arch design
Triton internal components
LibTriton.so


Android Application Build Process
.java files
Java Compiler
.class files
Obfuscator
Dx tool
Jar Signer
.so files
resource
Obfuscator
.apk files
APK Builder
.dex files
Ref: http://net.cs.uni-bonn.de/fileadmin/user_upload/plohmann/2012-Schulz-Code_Protection_in_Android.pdf


Resumed
(visible)
onResume()
onPause()
onResume()
Started
(visible)
Paused
(partially visible)
onStart()
onStop()
onStart()
Created
onRestart()
Stopped
(hidden)
onCreate()
onDestroy()
Destroyed