Securing Android

ASLR for Mobile Devices

Hristo Bojinov

Joint work with Dan Boneh and Google Android team

Motivation and Constraints

- ✔ Prevent return-to-libc attacks
- ✔ Complementary to..
  - Non-executable heaps
  - Non-executable stacks

- ✗ Limited disk footprint
- ✗ Limited processing capabilities

Android Platform Specifics

Prelinking

- speed up boot process
- save some space

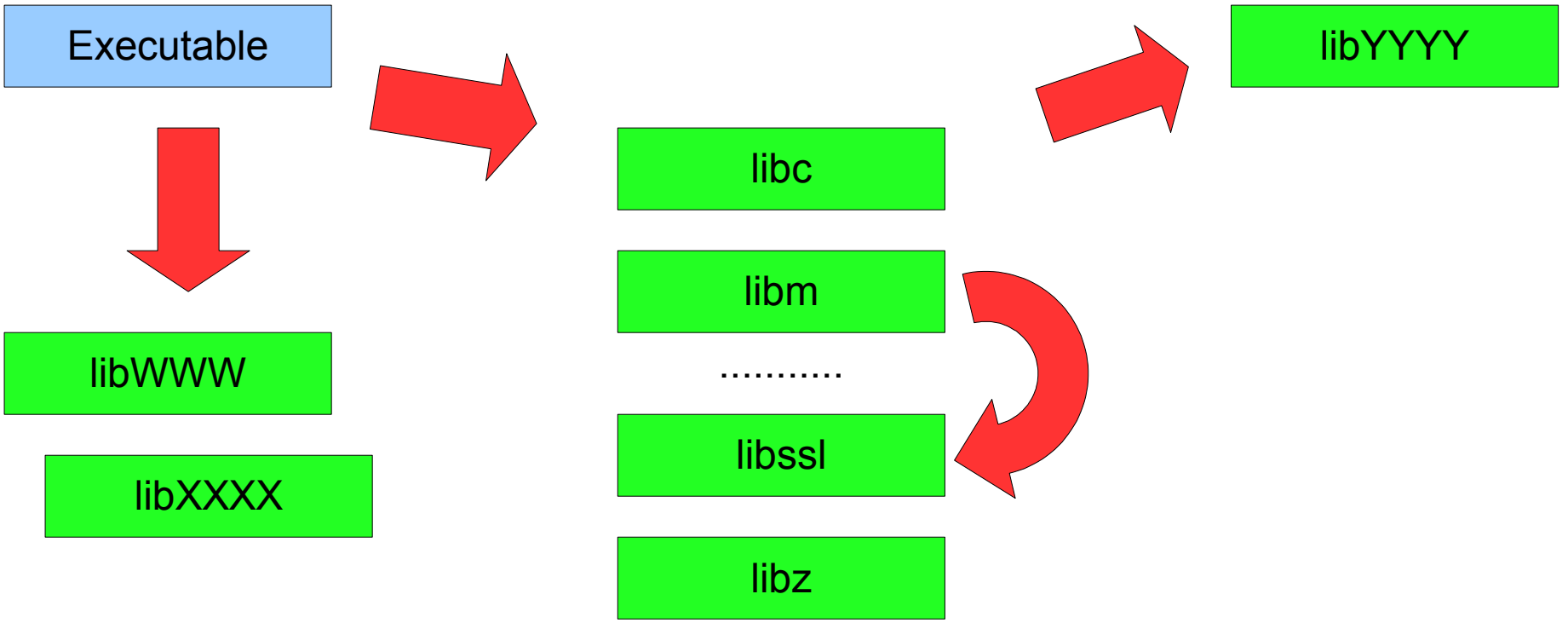
Update script language

- complete vs. incremental updates
- packaging manifest

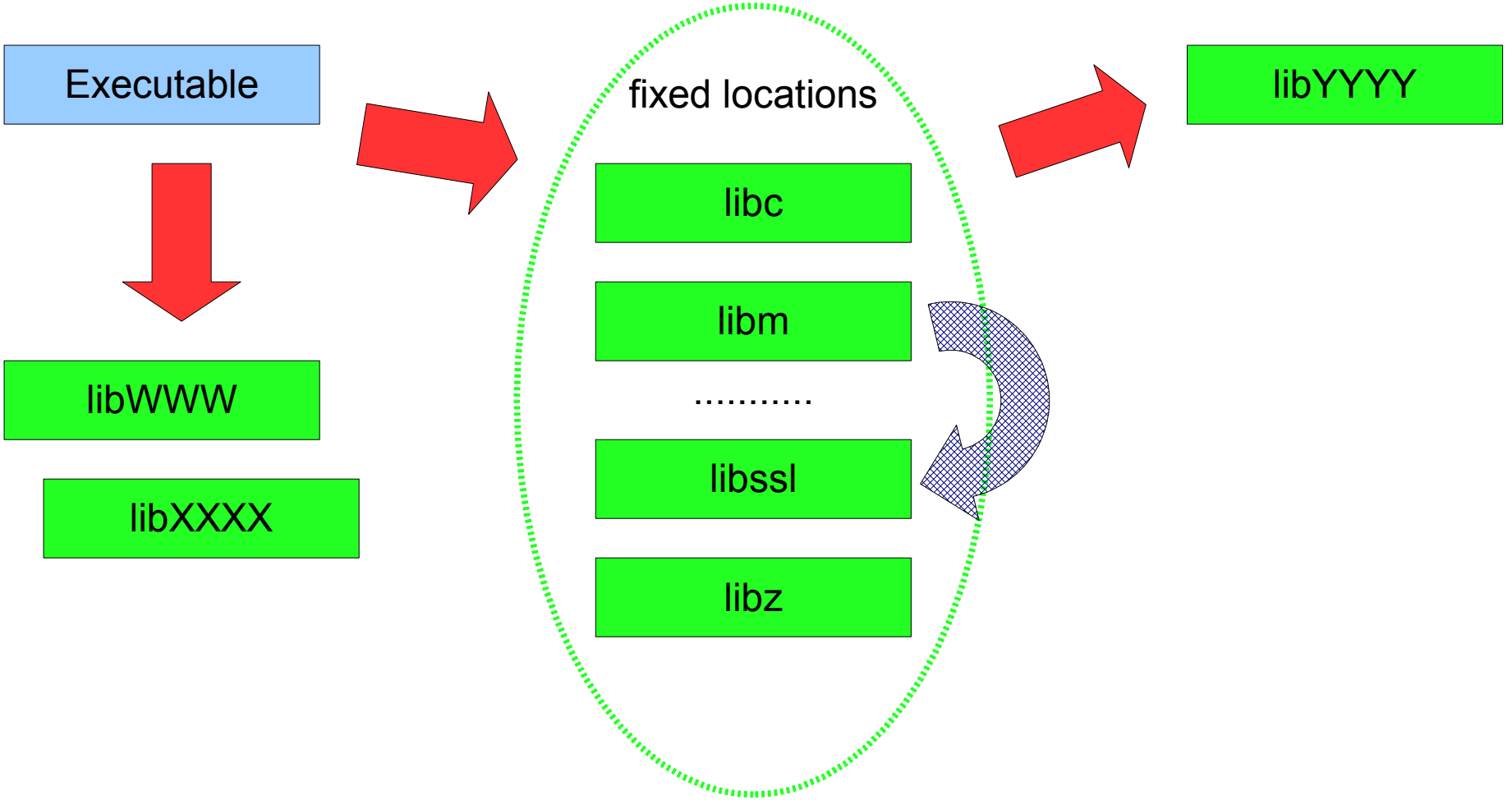
Update package validation

- e.g. check SHA-1 of file before patching

Dynamic Linking & Loading



Prelinking


Executable
libWWW
libXXXX
fixed locations
libc
libm
.....
libssl
libz
libYYYY

Our Approach

Surgical changes

- minimize kernel impact

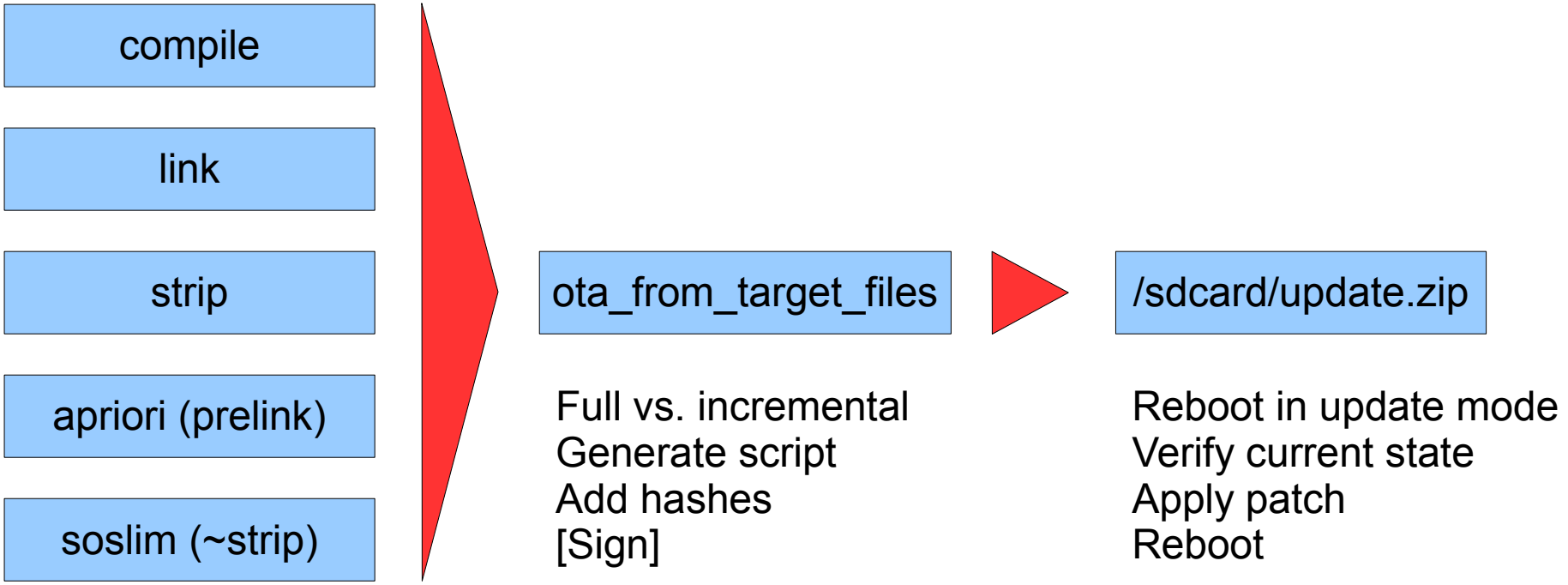
Preserve prelinking benefits

- some impact on footprint is unavoidable

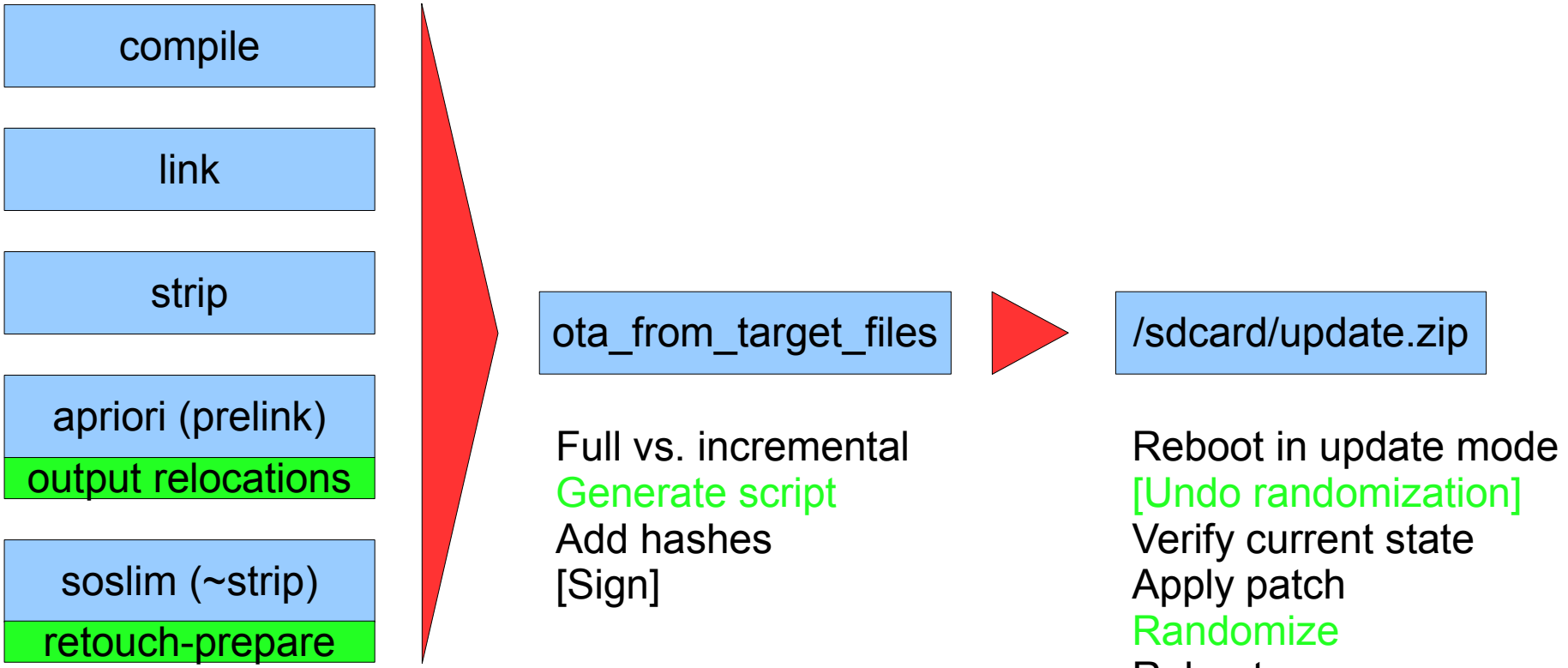
Apply randomization during update

- reduces risk; system files writeable
- almost as good as randomization at boot

Lifecycle of a Patch



A Patch with ASLR



Future Work

Add complemetary protection

- non-executable stack, heap
- randomize loader, base executables

Look at the whole platform

- login and user interaction
- application marketplace, permissions
- Java/Dalvik: security of JIT (e.g. w.r.t. JIT spraying)