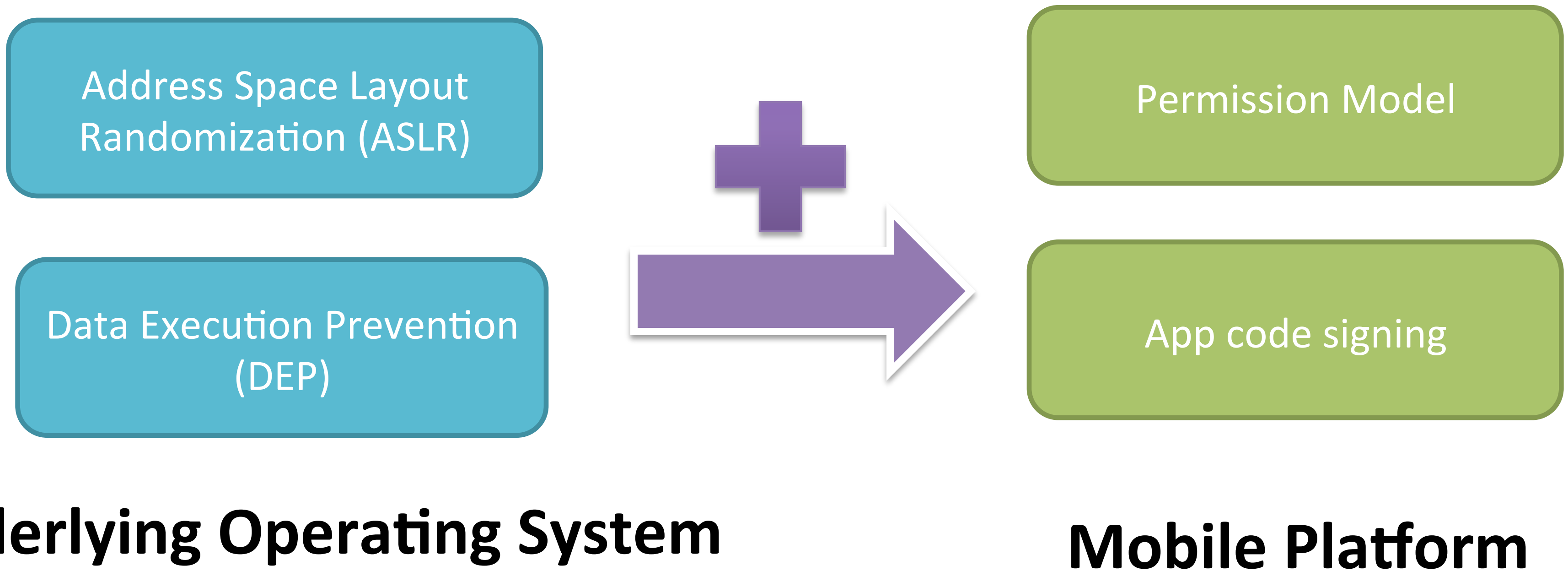
From Zygote to Morula: Fortifying Weakened ASLR on Android

Byoungyoung Lee[α]
Long Lu[β]
Tielei Wang[α]
Taesoo Kim[γ]
Wenke Lee[α]

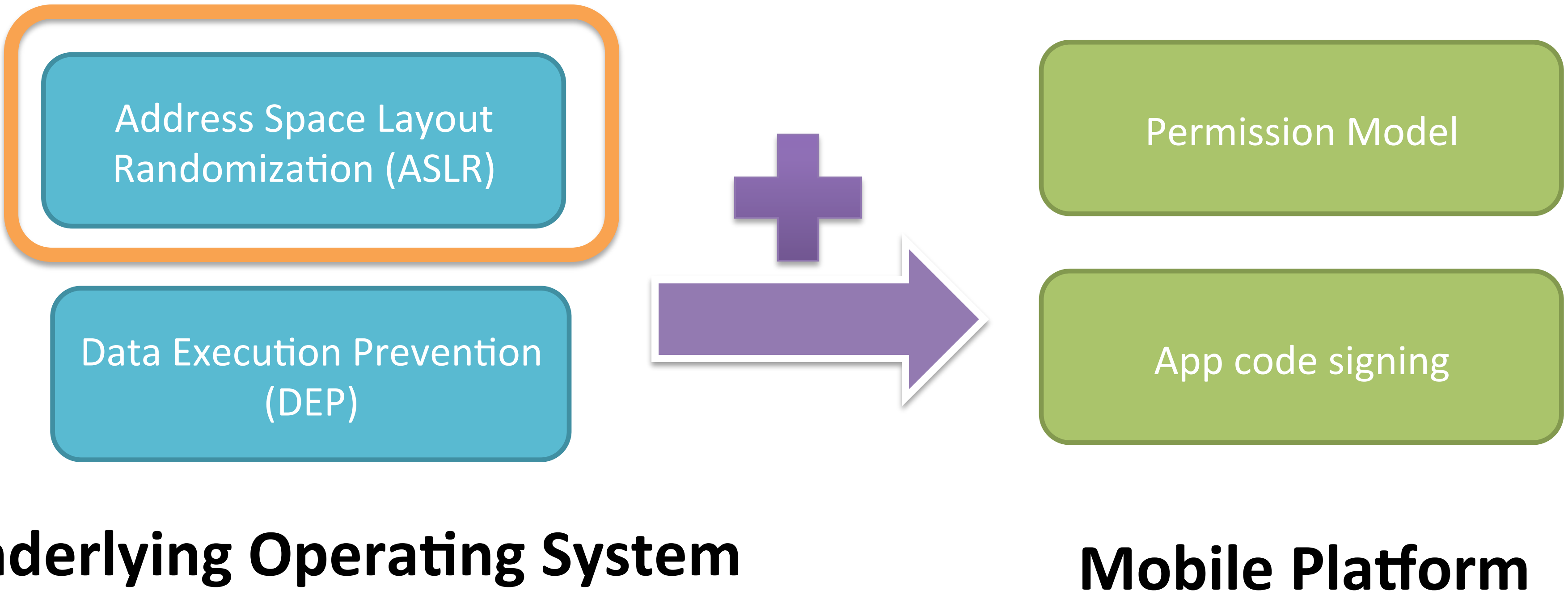
[α]Georgia Tech, [β]Stony Brook University, [γ]MIT

In embryology, the **morula** is produced by the rapid division of the **zygote** cell.
In Android, each application process is a fork of the **Zygote** process.

Security Hardening Efforts on Mobile


Address Space Layout Randomization (ASLR)
Data Execution Prevention (DEP)
Underlying Operating System
Permission Model
App code signing
Mobile Platform

Security Hardening Efforts on Mobile


Address Space Layout Randomization (ASLR)
Data Execution Prevention (DEP)
Permission Model
App code signing
Underlying Operating System
Mobile Platform

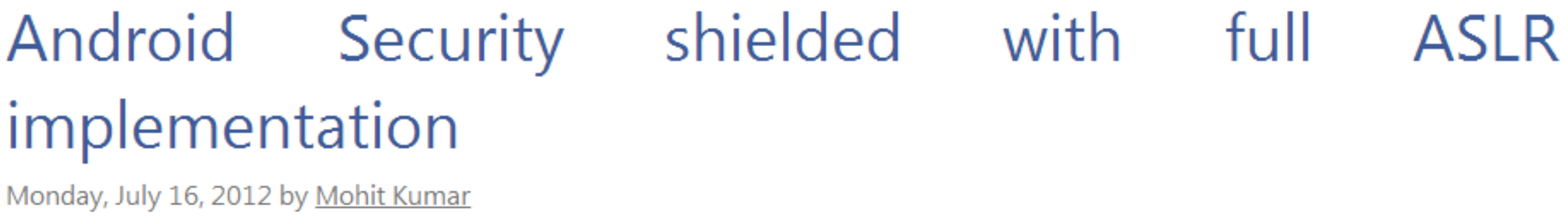
Address Space Layout Randomization (ASLR)

- Motivation
  - Knowing the address is prerequisite for many attacks
- Making prediction of the memory address difficult
  - – Individual memory layouts for each process
- Implemented in all major OSes

History of ASLR adoption in Android

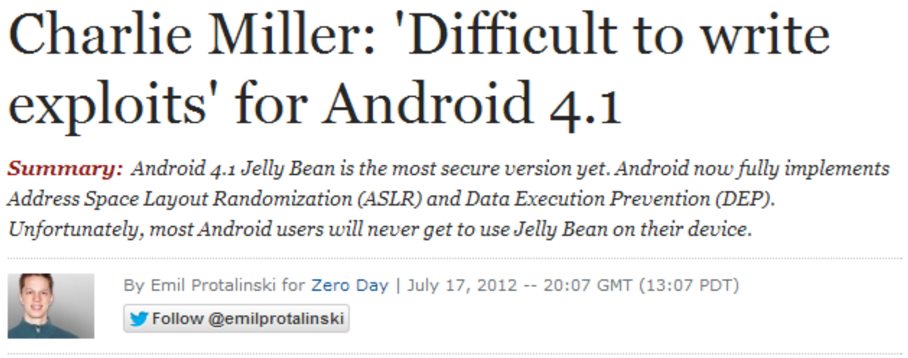
- Why ASLR on Android?
  - Prevent exploitations on native code in apps

- Adopted incrementally
  - Performance concerns on early Android devices (enabling PIE ➔ loading latency / memory overheads)
  - Android 4.1 implemented full ASLR enforcements



Android Security shielded with full ASLR implementation

Monday, July 16, 2012 by Mohit Kumar

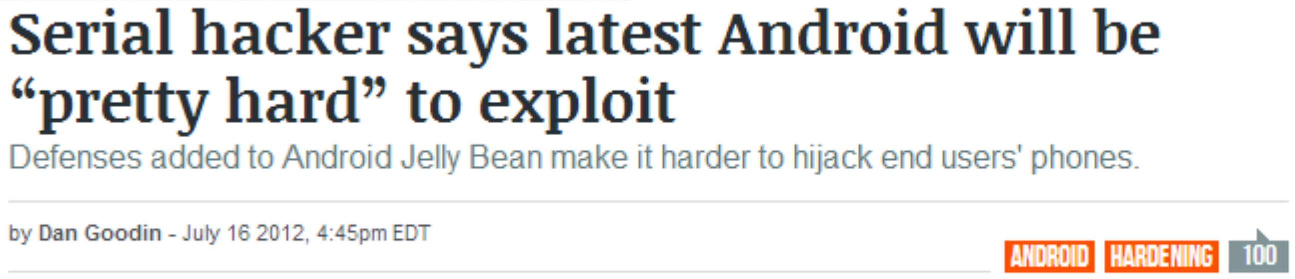


Charlie Miller: 'Difficult to write exploits' for Android 4.1

Summary: *Android 4.1 Jelly Bean is the most secure version yet. Android now fully implements Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP). Unfortunately, most Android users will never get to use Jelly Bean on their device.*

By Emil Protalinski for Zero Day | July 17, 2012 -- 20:07 GMT (13:07 PDT)

Follow @emilprotalinski



Serial hacker says latest Android will be "pretty hard" to exploit

Defenses added to Android Jelly Bean make it harder to hijack end users' phones.

by Dan Goodin - July 16 2012, 4:45pm EDT

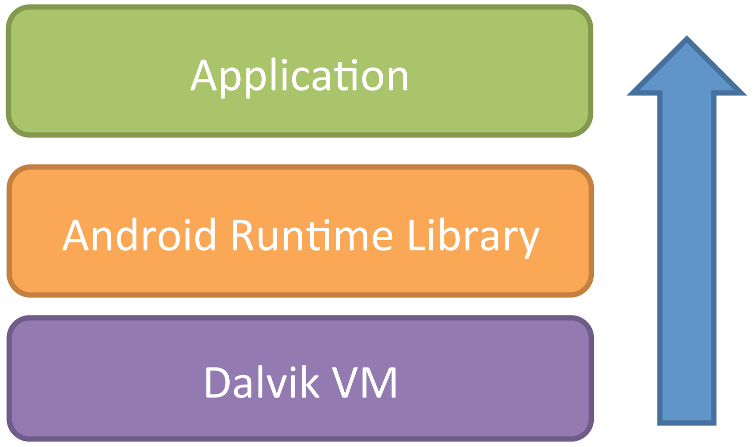
ANDROID HARDENING 100

(actual) ASLR enforcements in Android related to performance prioritized design

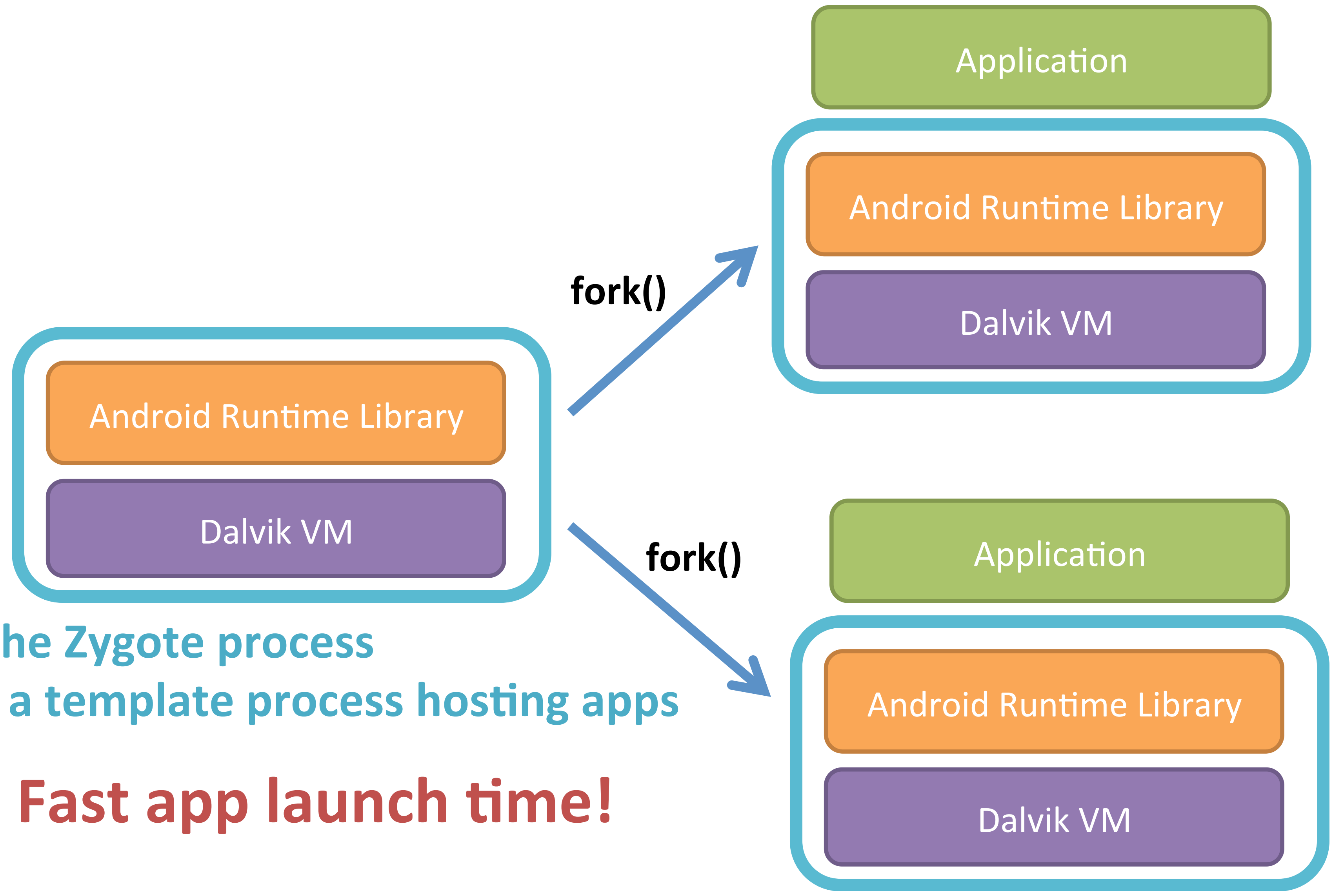
Performance Prioritized Designs of Android

- Multi-layered architectures
  - Android Applications run on Dalvik VM
  - with additional runtime libraries

➔ Slow app launch time



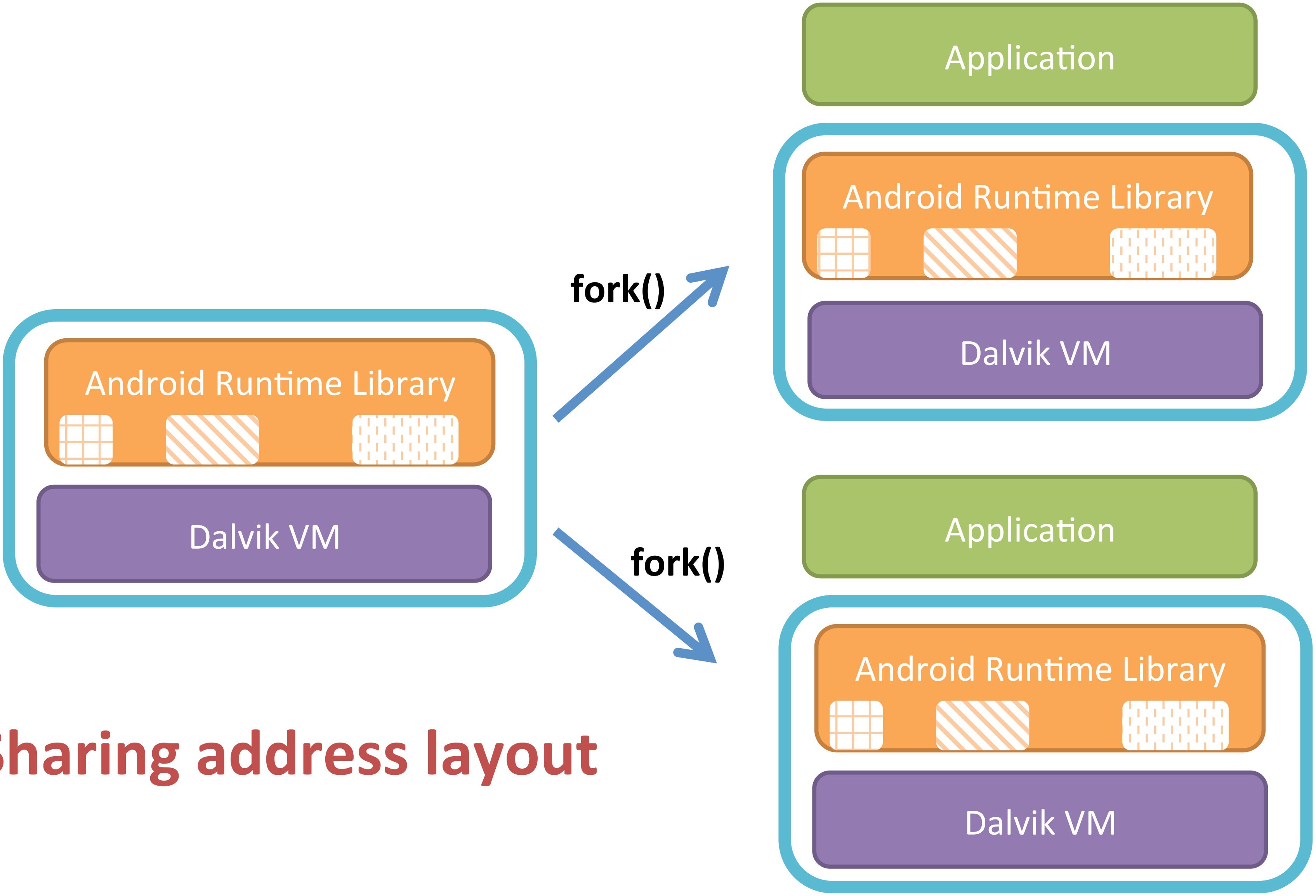
Zygote: the process creation module



the Zygote process
: a template process hosting apps

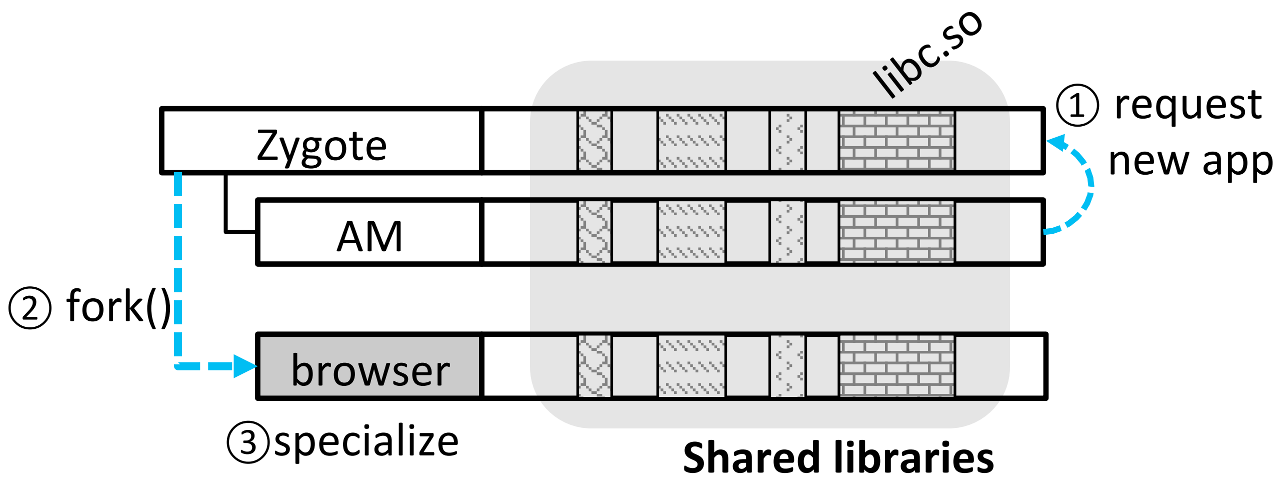
Fast app launch time!

Zygote: the process creation module



Sharing address layout

Zygote weakens ASLR effectiveness



- All apps have the same memory layouts
  - For shared libraries loaded by the Zygote process

→Weakens Android ASLR security

Attacking the ASLR weakness by Zygote

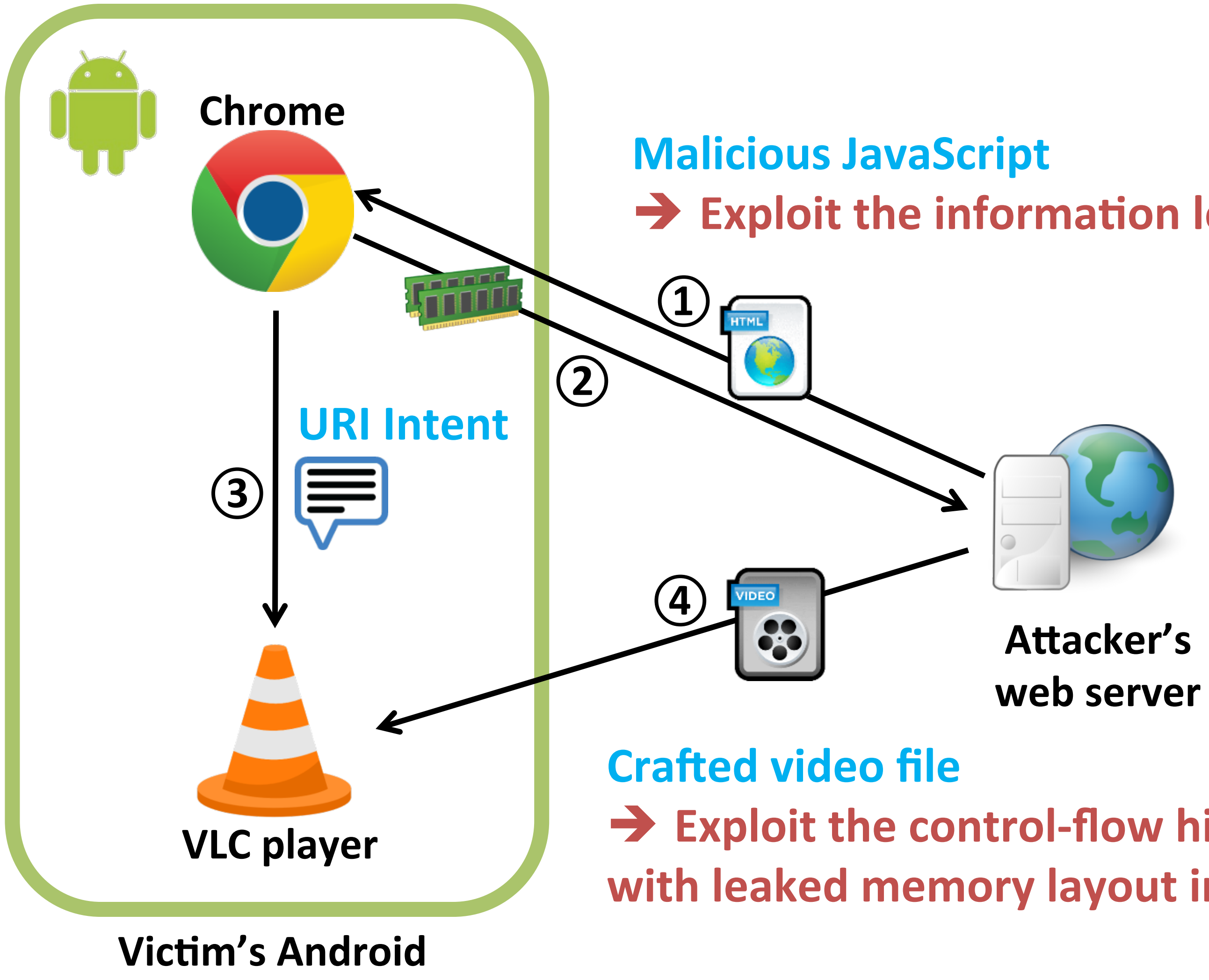
- Challenges to develop fully working exploits (with ideal ASLR)
  - Exploit the **Information leak** vulnerability
  - Exploit the **control-flow hijack** vulnerability

  ➔ should be achieved in **the same app**!

Attacking the ASLR weakness by Zygote

- How Zygote’s ASLR weakness helps attackers
  - **Remote Coordinated Attacks**
    - Information leak in Chrome + control-flow hijack in VLC
    - Reduce the vulnerability searching spaces
  - **Local Trojan Attacks**
    - Obtain the memory layout by having the trojan app installed

Attacking weakened ASLR : Remote Coordinated Attack



Malicious JavaScript
➔ Exploit the information leak vulnerability (CVE-2013-0912)

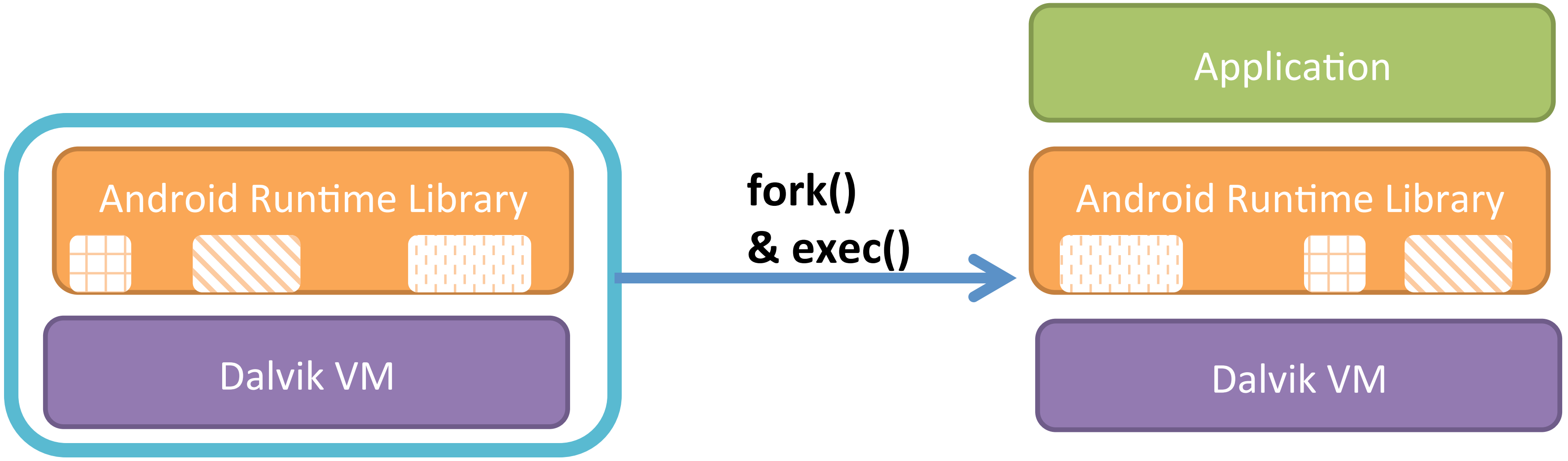
Crafted video file
➔ Exploit the control-flow hijack vulnerability with leaked memory layout information

Attacking weakened ASLR : Local Trojan Attack

- Zero-permissioned trojan app
  - Asks (almost) no permissions to the system
  - Scanning memory spaces using the native code
  - Layout information can be exported
    - Intent
    - Internet

- Once the trojan app is installed, ASLR can be easily bypassed

Intuitive (but impractical) Solutions

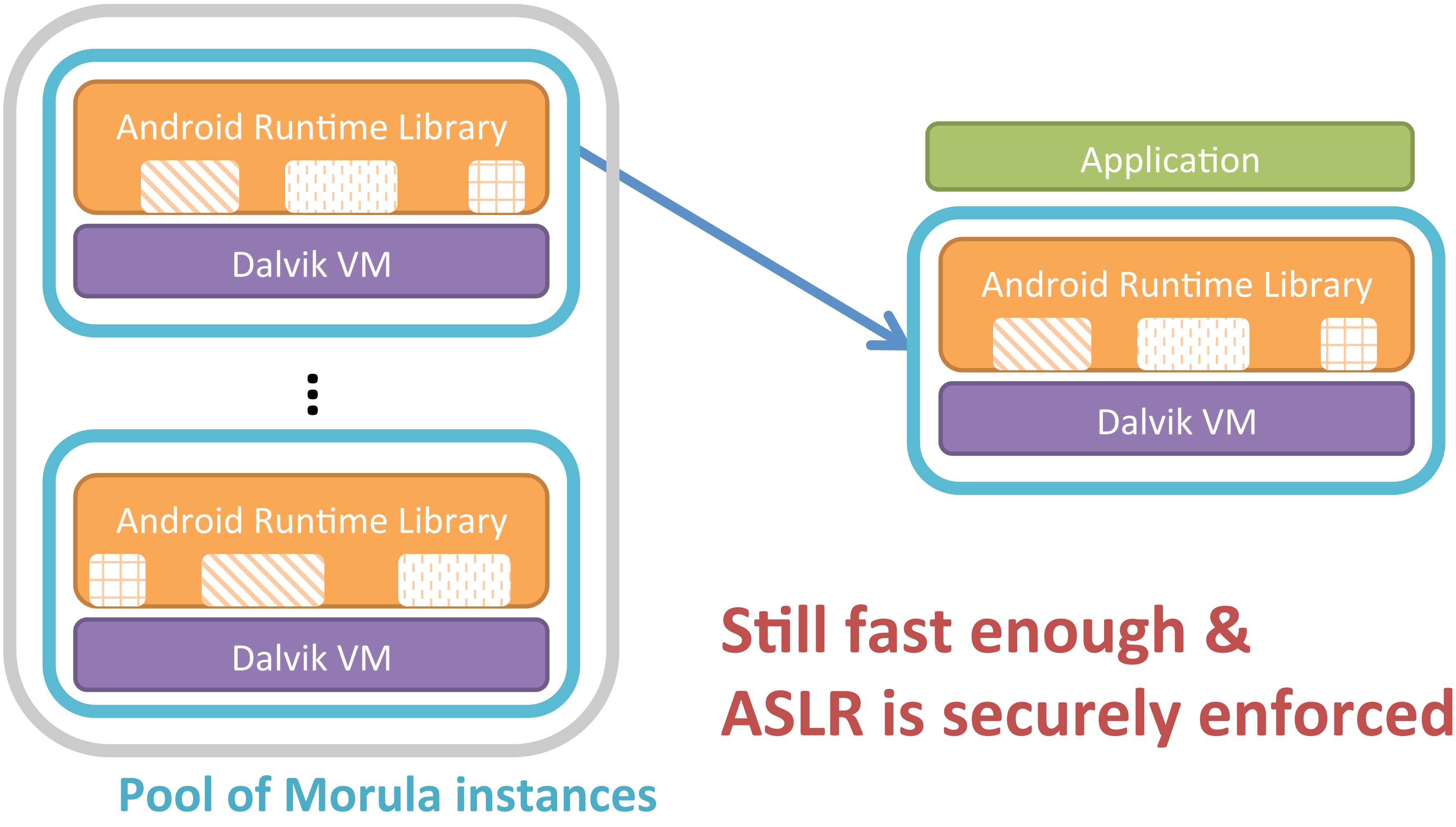


- fork() & exec()
  - Execute and initialize all components from the scratch
- Too slow to be used in practice
  - App launch time: 427% slowdown

Morula: Fast Process Creation without Weakening ASLR

- Maintain a Morula instance pool
  - An instance is prepared (fork() and exec()) when the device is idle
  - Pull out the instance to create an app later

Morula: Fast Process Creation without Weakening ASLR

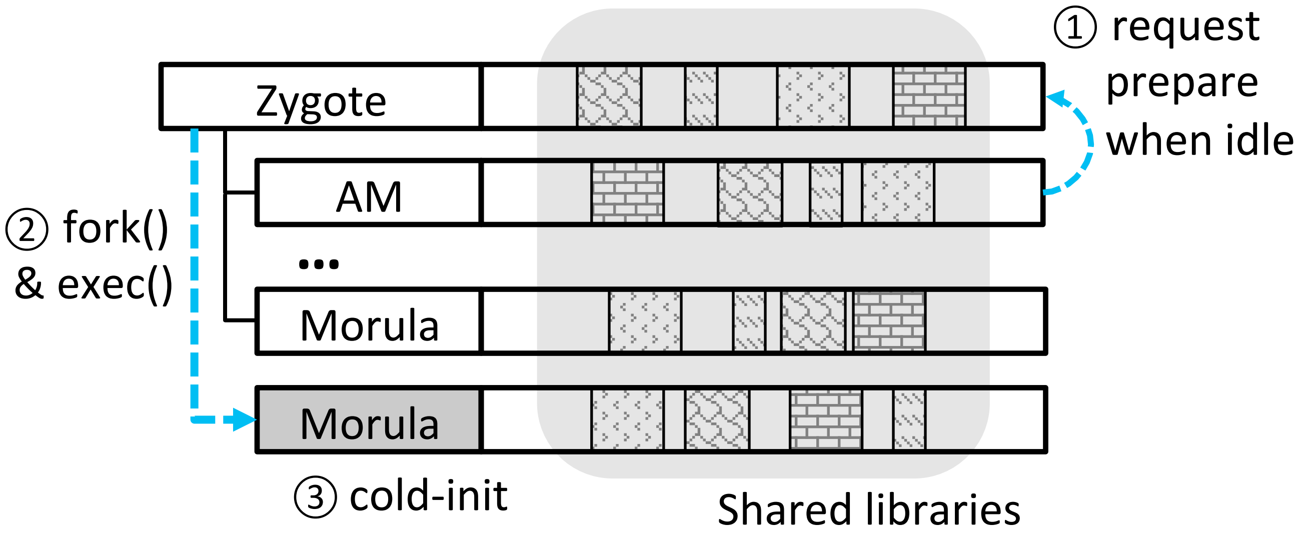


Still fast enough &
ASLR is securely enforced

Morula: Fast Process Creation without Weakening ASLR

Preparation phase

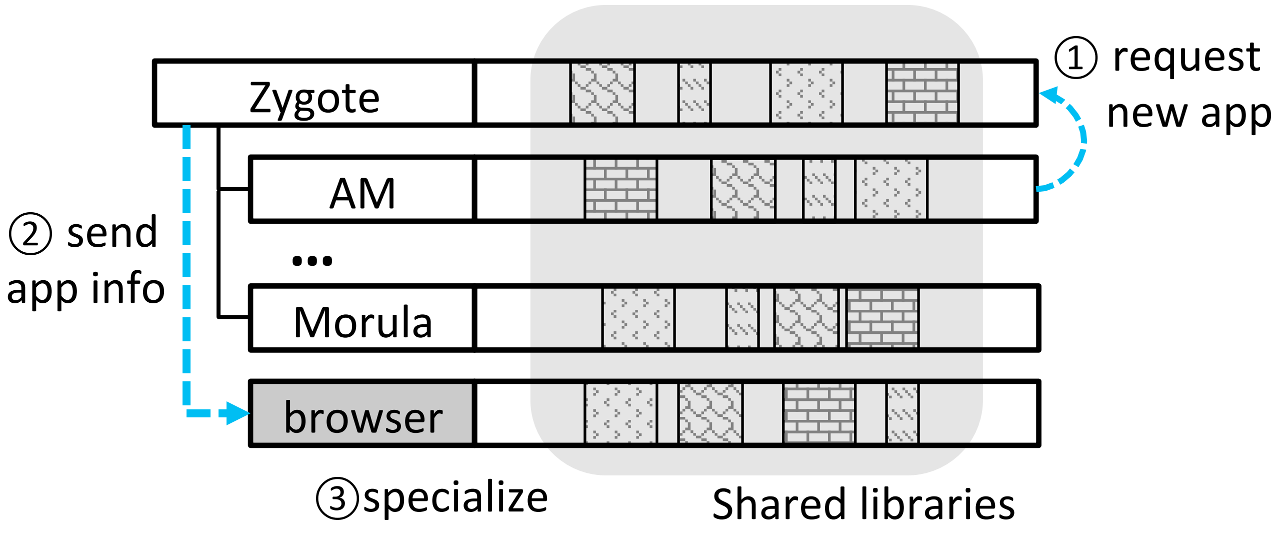
– Prepare a Morula instance when the device is idle



Morula: Fast Process Creation without Weakening ASLR

Transition phase

– Transform the instance into the target application

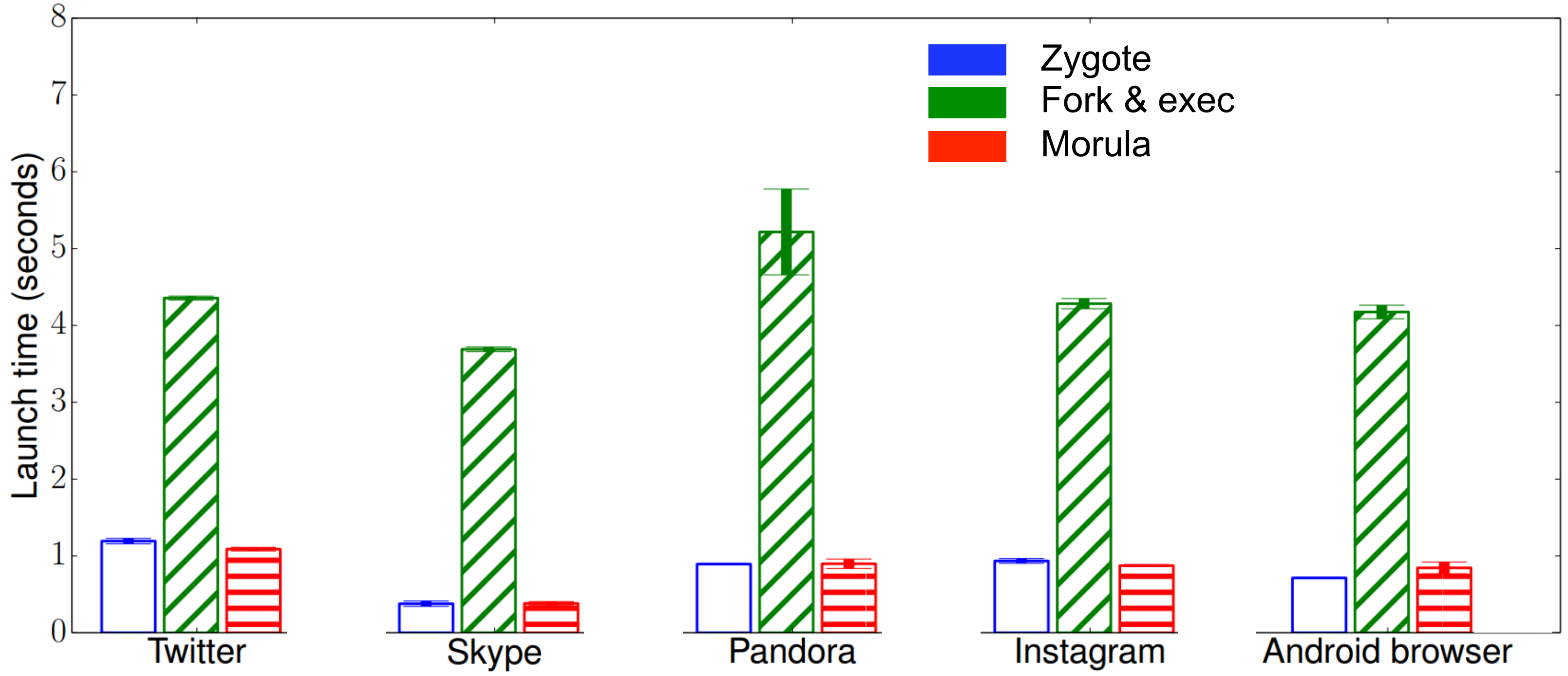


Evaluations

- Implemented Morula in Android 4.2
  – 548 Loc in Java
  – 197 LoC in C

- Evaluated on Galaxy Nexus
  – Dual-core 1.2 GHz CPU, 1GB RAM

Application Launch Delays

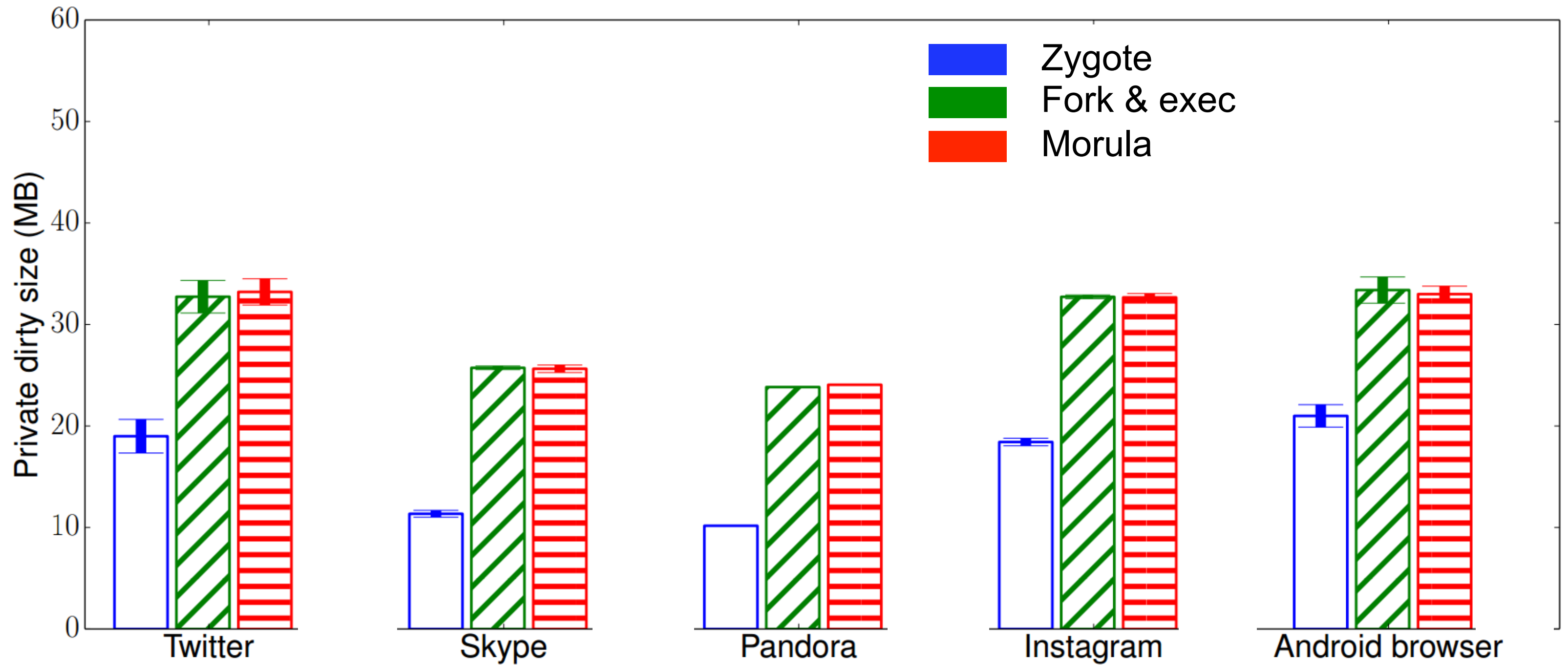


Morula is 0.7% faster than Zygote on average

➔ Trade-offs between

fork() in Zygote VS extra communications in Morula

Memory Use Overheads



Morula uses 13.7MB (85%) more memory per app
➔ Mostly from individual DalvikVM heaps

Implications

- Other data sharing issues by Android Zygote
  - Predicting OpenSSL's PRNG states [CCS 13]
- Systems relying on Zygote like designs
  - Chrome, Server side applications [Blind ROP, Oakland 14]
  - Platform specific optimizations should be considered
    - Morula: On-demand Dalvik class loading / Selective randomization
- Performance prioritized designs can be odd with ASLR
  - Hash table designs in dynamic script languages (to appear, BlackHat USA 2014)

Conclusion

- Showed two attacks on Weakened Android ASLR by Zygote
  - Remote coordinated attacks
  - Local trojan attacks

- Morula achieves fast app launch time without weakening ASLR

Thank you

Q&A

Backup slides

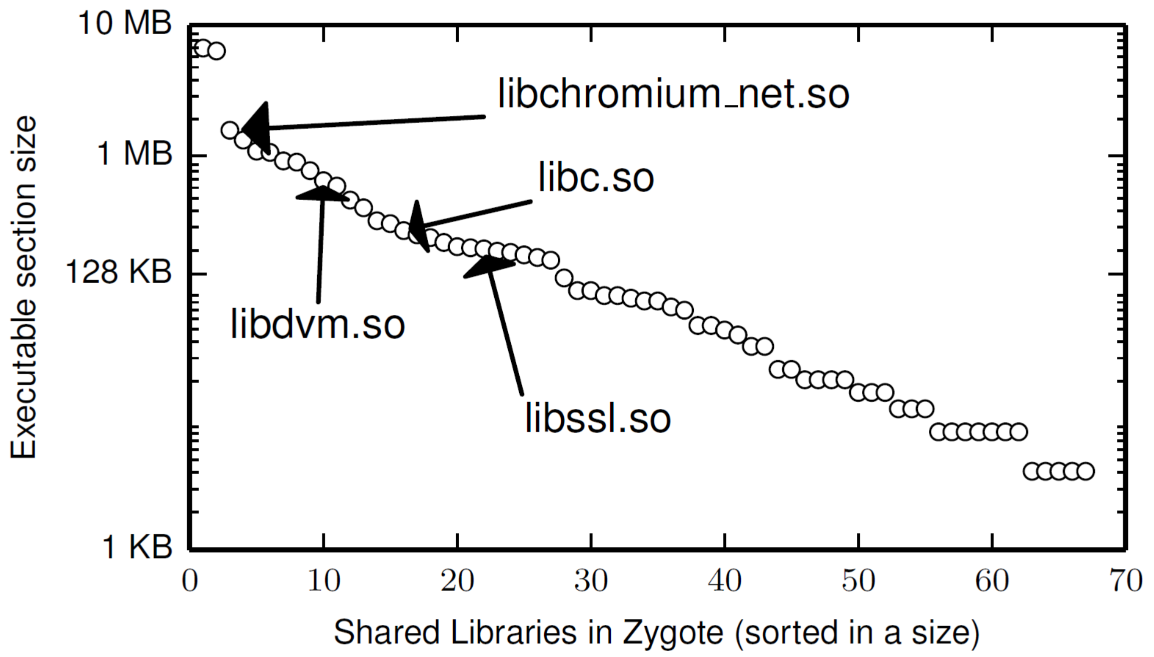
Battery Consumption

- Executed the web browser every 10 seconds
- Morula imposes 0.5% more power consumption for 100 executions

Optimizations on Morula

- **On-demand Dalvik class Loading**
  - DalvikVM pre-loads 2,541 Dalvik classes
    - Most Java libraries including Android runtime libraries
    - On average, only **5%** of them are actually used by applications
  - Dynamically loaded the classes at the time it is used
    - Shift the overheads from preparation phase to transition phase
  - Device booting process can be significantly benefited
    - More than 10 apps are launched during the device booting

Zygote conflicts with ASLR security



➔Each app shares ~10MB sized of data that can be potentially used as attack vectors (e.g., ROP)