RetroSkeleton: Retrofitting Android Apps

Benjamin Davis, Hao Chen
University of California, Davis

MobiSys 2013

The Android™ Platform

More than 600,000 third-party apps & games on Google Play™ alone

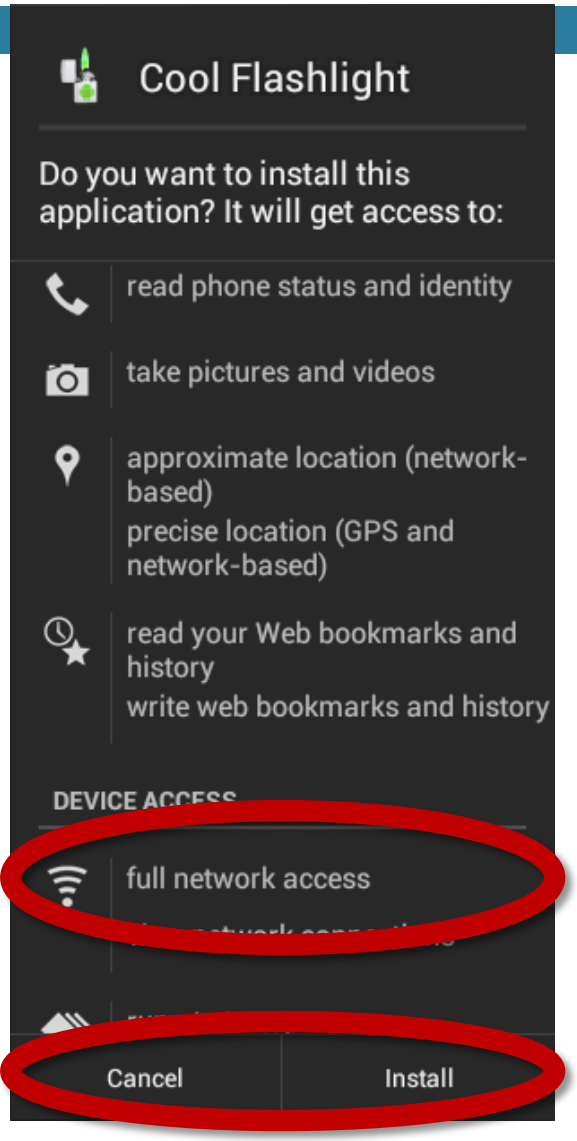
?

“What are these apps doing?”

“How can I control what these apps do?”

Limitations of Android's Permission System

- All or Nothing
- Coarse-grained
- No revocation



Current Proposals: Platform Modifications

- Examples:
  - TISSA (TRUST ‘11)
  - Apex (ASIACCS ‘10)
- Deployment Challenges
  - Proprietary binaries for device hardware
  - Requires rooting phone, voiding warranty, etc.
- Inflexible
  - Difficult to enforce app-specific policies
  - New behavior requires system changes
  - Each Android version requires new implementation

Goals

- *Observe and control the behavior of third-party apps*

- Devices
  - Require no platform modifications
- Approach
  - Powerful
  - Complete
- Policy
  - App-independent
  - Applied automatically

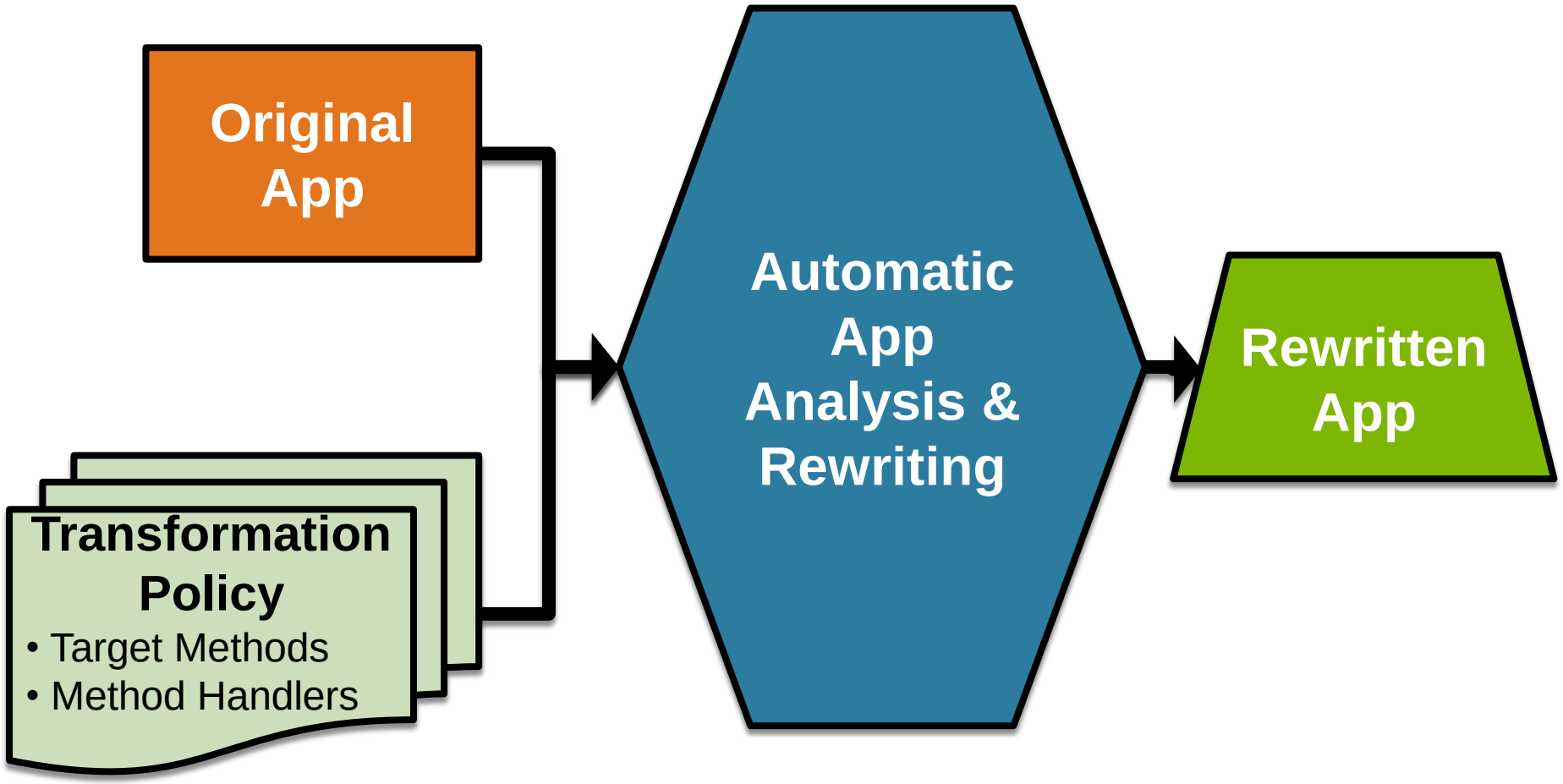
- Non-goal: prevent detection by app

Rewriting Android Apps

- Observations:
  - Apps interact with device via platform API method calls
  - 95% of apps are implemented entirely in Dalvik
  - Dalvik bytecode is structured & unambiguous

- Our approach: **in-app method-call interception via automatic bytecode rewriting**

RetroSkeleton

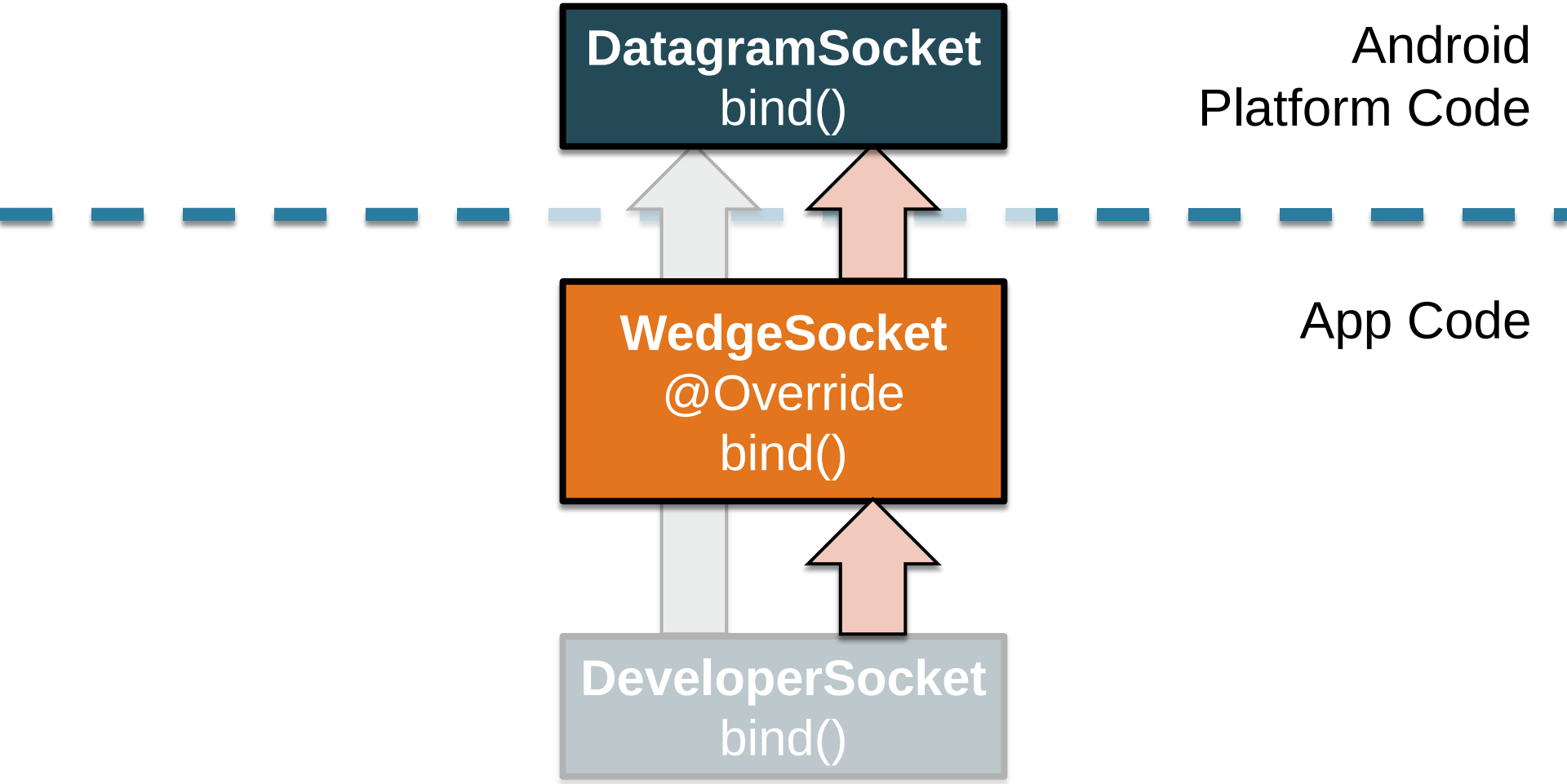


Intercepting Method Invocations

- Interception strategy depends on:
  - Method type (`static`, `instance`, …)
  - Method attributes (`protected`, `final`, …)
  - Invocation kind (`direct`, `virtual`, …)

- Higher-level strategies for:
  - Inheritance
  - Virtual method invocation

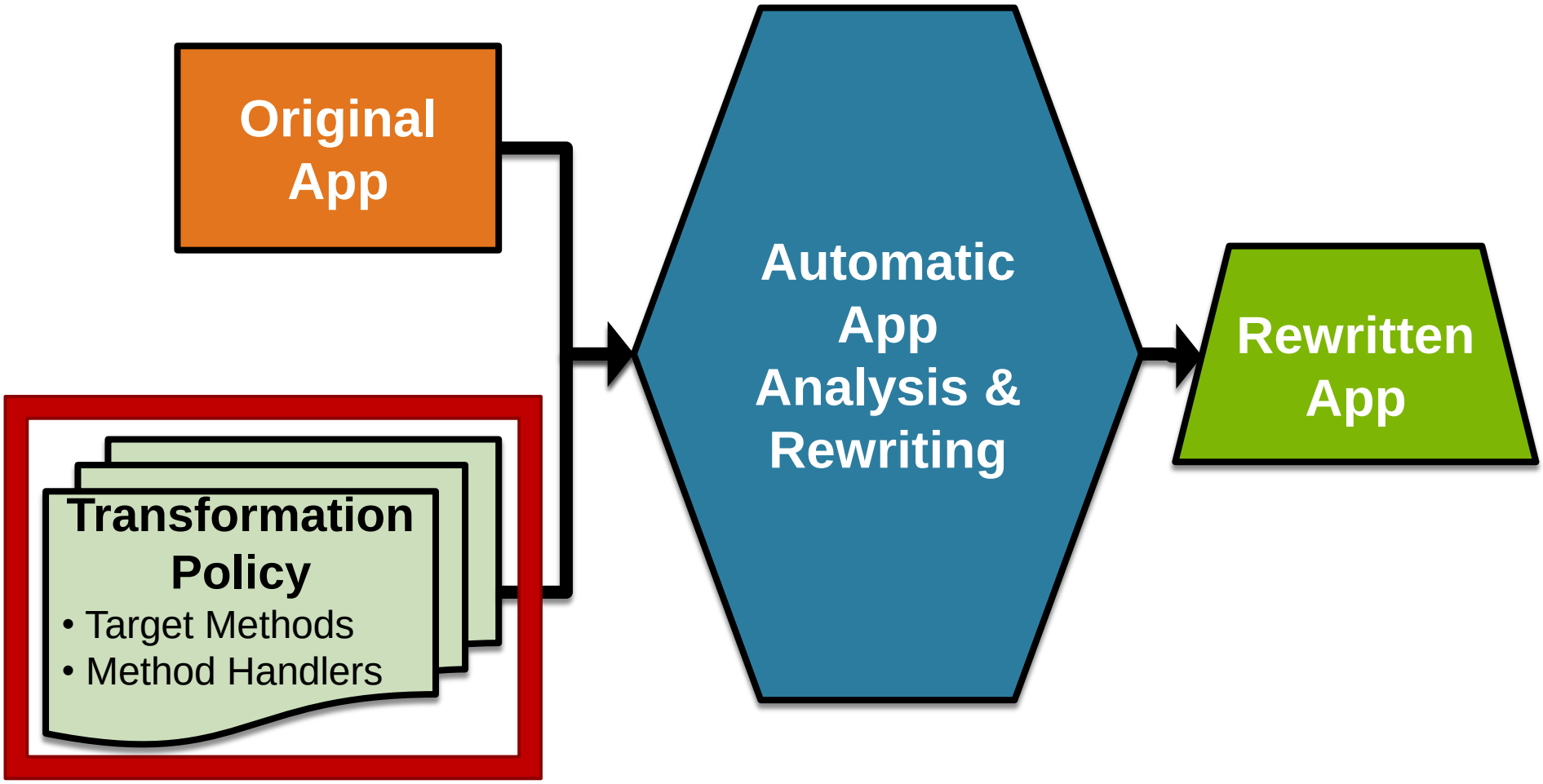
Inheritance-Based Interception



Challenge: Interception Completeness

- Reflection API (behavior specified at runtime)
  - Statically identify **invocation** of the reflection API
  - Add handlers to inspect and dispatch at **runtime**
- Native and dynamically-loaded code
  - Detect and intercept invocation

RetroSkeleton



Transformation Policy Specification

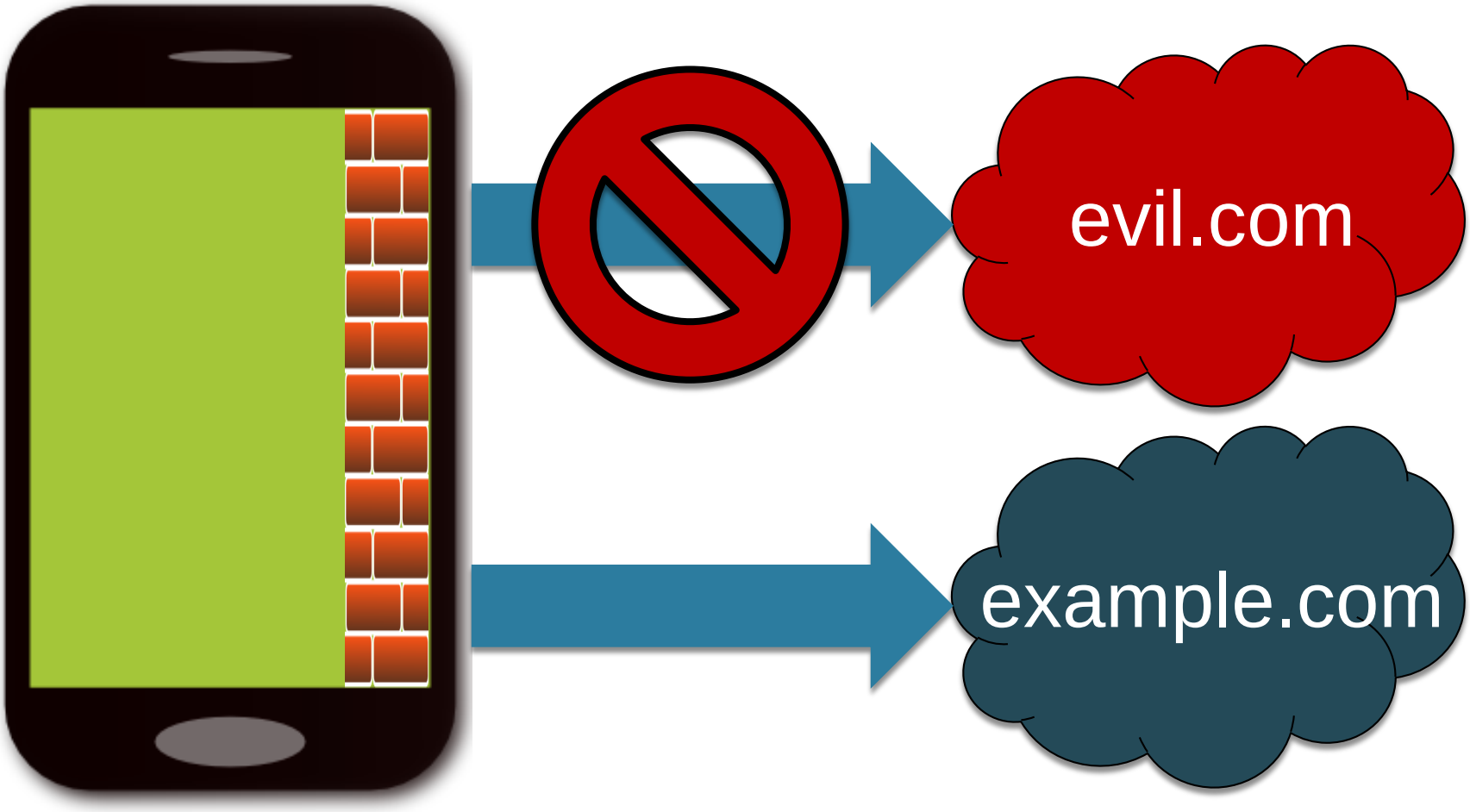
- **Target Method**

```
java.net.DatagramSocket
    public void connect(SocketAddress peer)
                throws SocketException
```

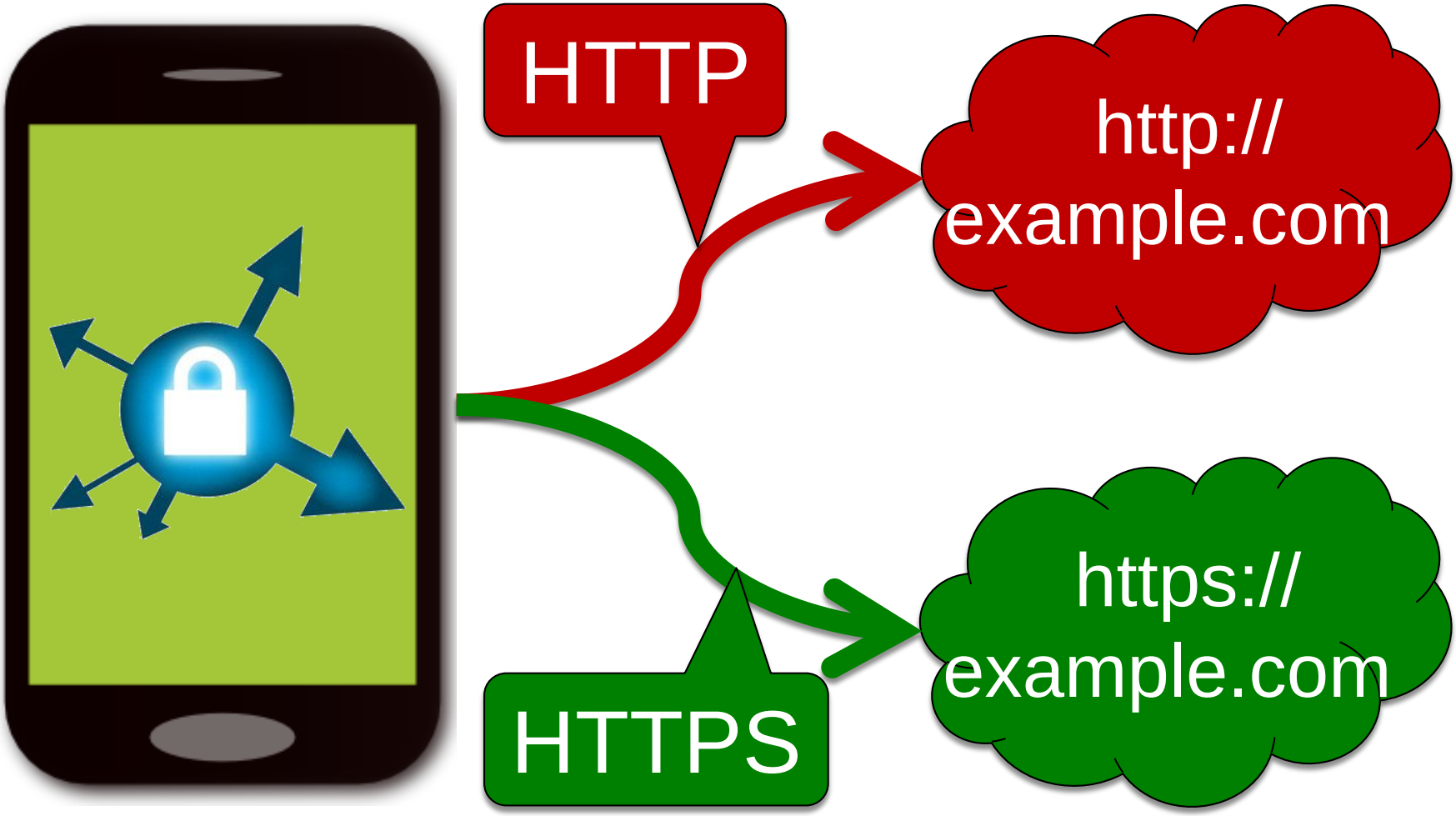
- **Handler Behavior**

```
public static void retroSkeletonConnect
  (DatagramSocket p0, SocketAddress p1)
                    throws SocketException
{
     Log.i("RSKEL", "connect called!");
     p0.connect(p1); // invoke target method
}
```

Use: Fine-Grained Network Access Control

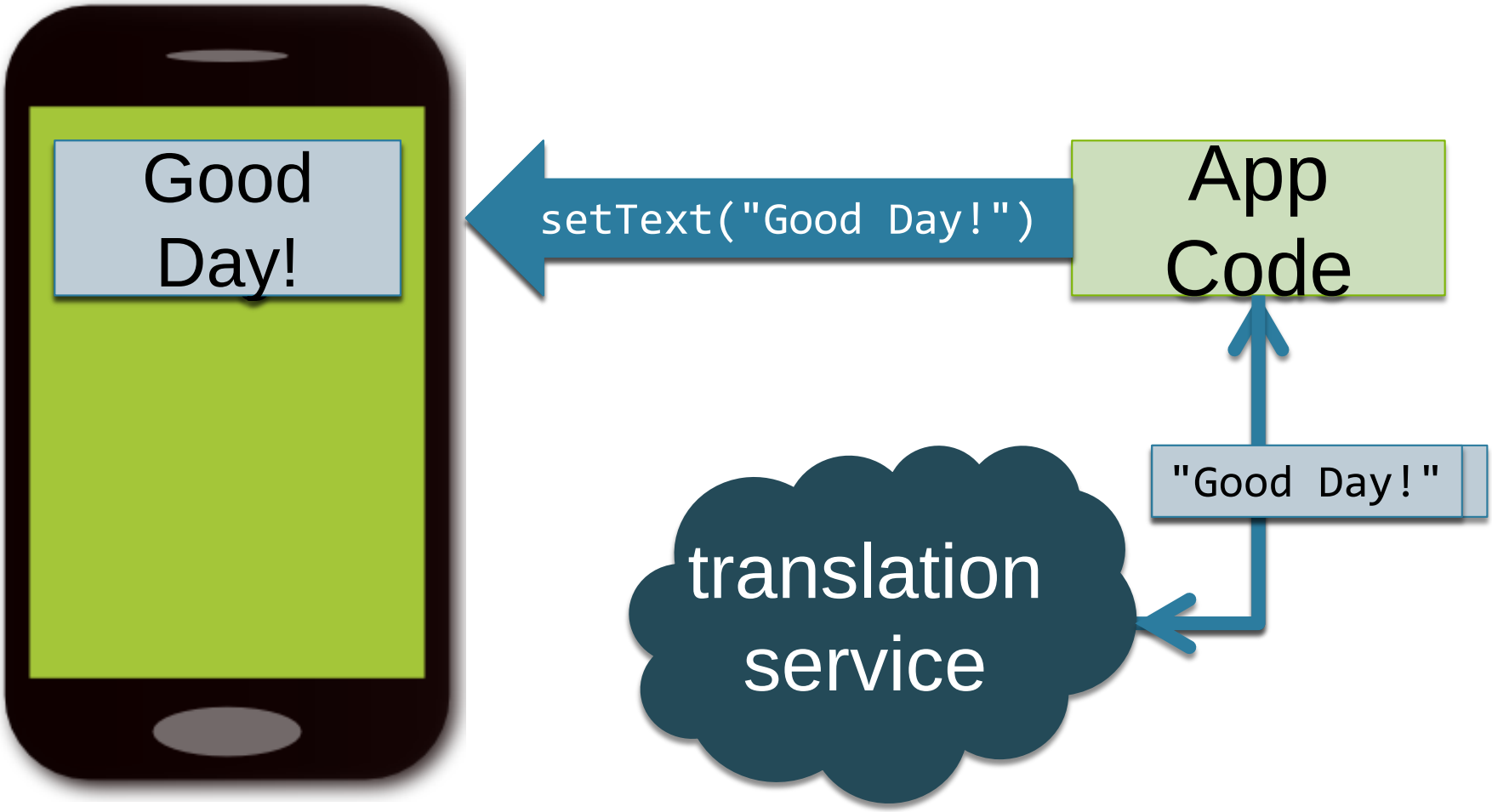


Use: HTTPS-Everywhere for Apps



HTTPS Everywhere Project: https://www.eff.org/https-everywhere

Use: Automatic App Translation



Evaluation

- Run-time overhead: (0.2 µs) + handler
- Rewrite speed: fast (~5 seconds)
- Impact on app size: (0.5% for our policies)
- Policy functionality:
  - Applied to over 1,000 apps from Google Play
  - Tested rewritten apps in emulator & observed handler behavior

Transformation Policy	Success
Network Access Control	99.5%
HTTPS-Everywhere	93.2%
Automatic Translation	99.6%

Conclusion

- In-app method interception
- App-agnostic policy specification and application
- Automatic rewriting, no manual guidance
- Rewritten apps deployable to any Android device