BYOD for Android — Just add Java

Jessica Buttigieg, Mark Vella, and Christian Colombo
[jessica.buttigieg.12][mark.vella][christian.colombo]@um.edu.mt

UNIVERSITY OF MALTA
L-Università ta' Malta

Motivation

In a Bring Your Own Device (BYOD) setting employees use their personal mobile devices to access enterprise resources. This poses a security concern where un-trusted user-installed applications might interfere maliciously with corporate ones. Android has limited support for dual work-personal contexts that either outright excludes non-work apps or require apps to be programmed specifically for BYOD. Other solutions focus on malware scanning and virtualization. A dynamic policy system can further benefit BYOD in providing *both* dynamic permissions and context-specific app functionality, without offloading security-critical decisions to device users.

Contributions

Runtime Verification (RV) – Context-based policy rule definition revolving around Android Java API.

Lightweight **Dynamic Binary Instrumentation** (DBI) – Hooks Android API methods without modifying Android/app source code.

BYOD-RV: First-stage experimentation

RV

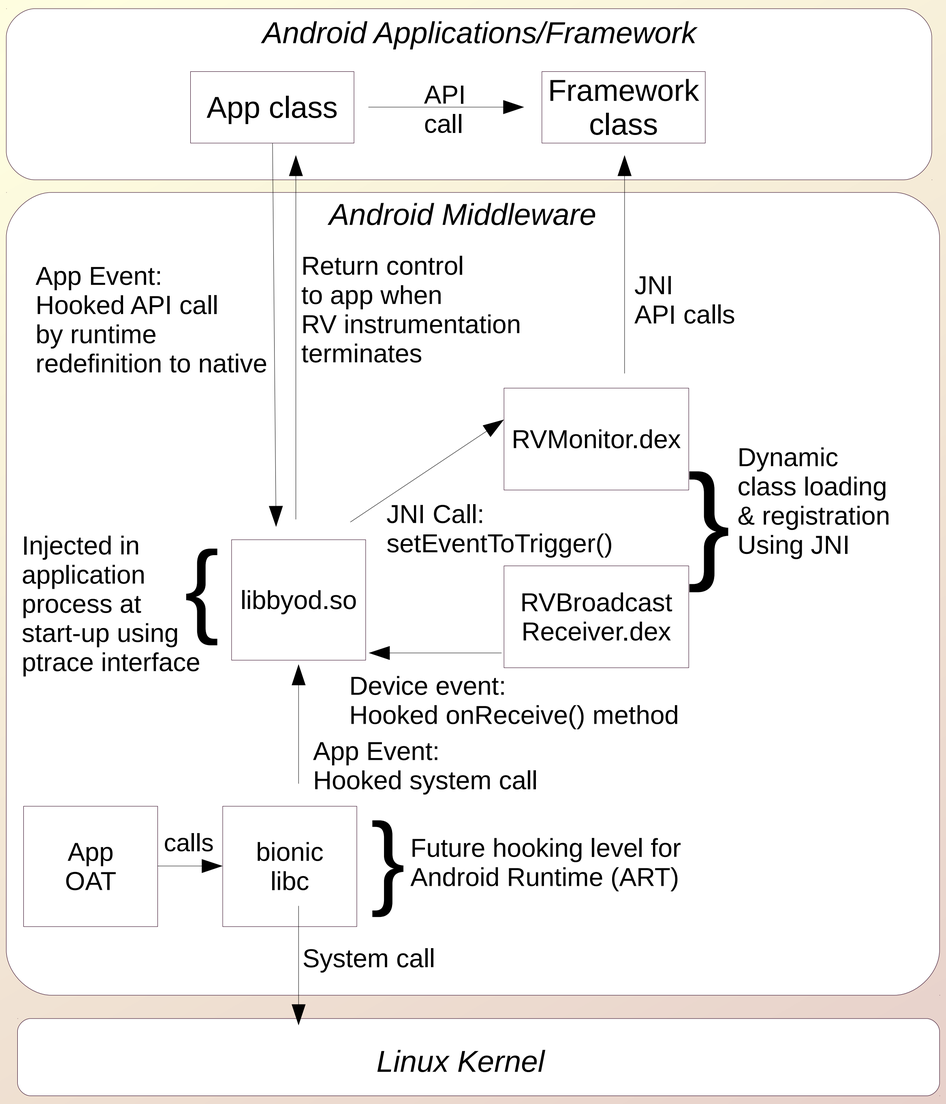
Rules follow a Guarded Command Language format:

`Event | Condition → Action`

and are implemented in Java with access to Android API calls.

```
class RVMonitor {

    public RVMonitor() {
        global_rs1 = DeclareRuleSet1(); ….
    }

    public static RuleSet DeclareRuleSet1() {

        class RuleSet1 extends RuleSet {public boolean event1; ….}

        final RuleSet1 rs1 = new RuleSet1();
        rs1.addRule(new Rule("ARule") {public void condition(){...} public void action(){...} }); …
        return rs1;
    }

    public void setEventToTrigger(String EventName) {
        rs1.trigger(EventName); … //eventually calls individual rule condition()/action() methods
    }
}
```

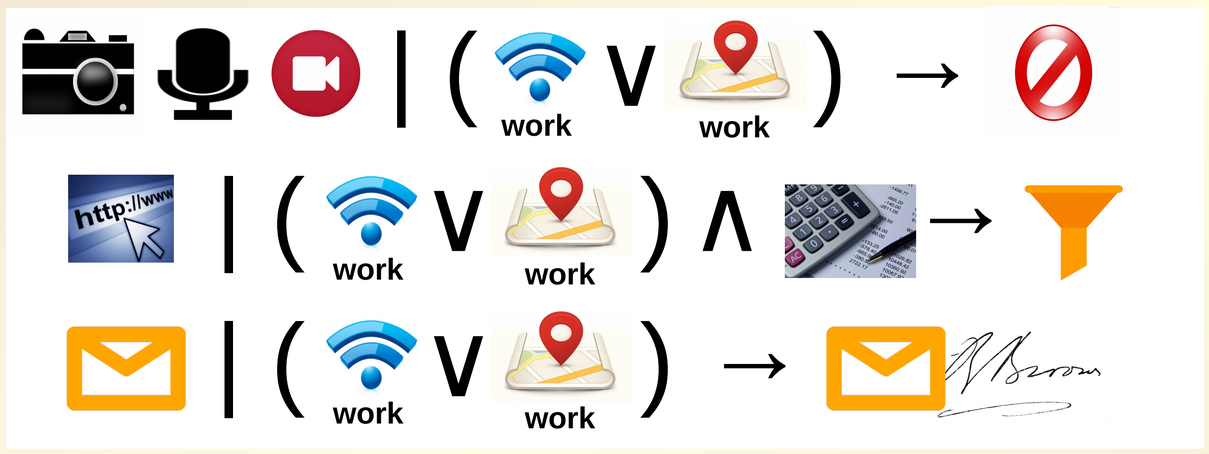
DBI



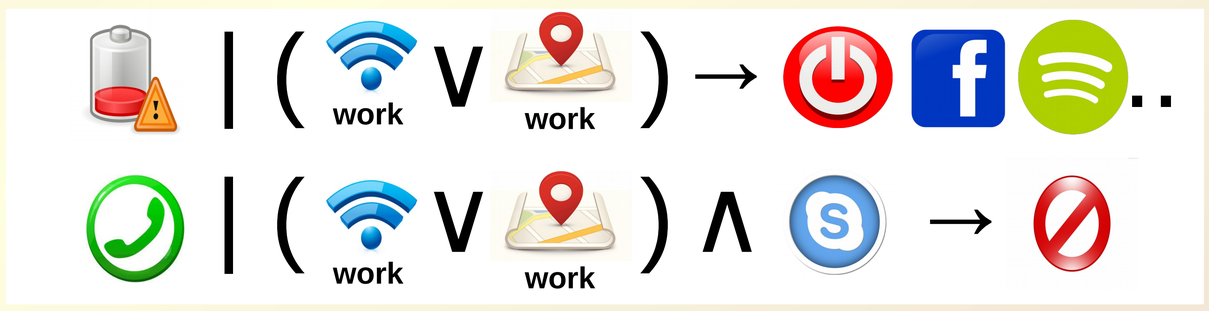
Case studies

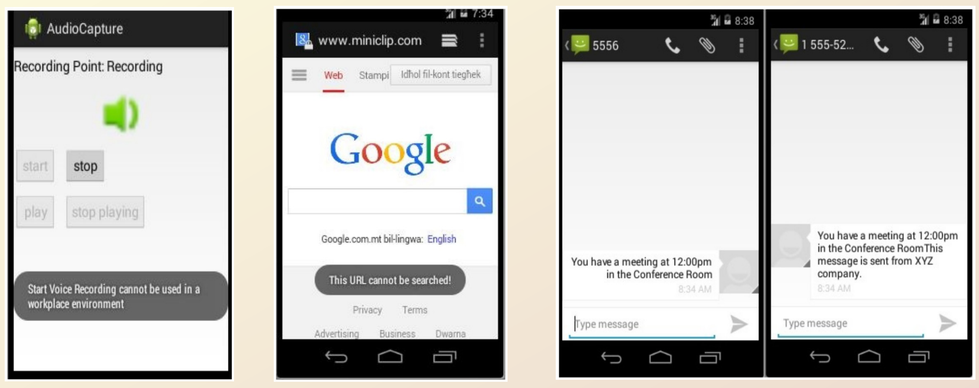
Prototype implementation based on the DDI toolkit.

Application-level Events



Device-level Events





Future challenges

RV: *Domain-Specific Language (DSL) provision for defining policies*
- Move away from Java to a more natural way to define rules.
- Decouple policy definition from enforcement.
- Requires DSL design & compilation. Requires a-priori hooking of all security-critical events.

DBI: *Single central RV monitor & Port to ART*
- Requires separating event collection and monitoring.
- Compiled OAT files only allow for system call-level hooking, introducing a semantic gap challenge.
- JNI-driven dynamic class loading and access to the Android API from RVMonitor is still possible through ART-mediated JNI.

Practical deployment: *Runtime overheads and Deployment model*
- Battery life and retaining prompt application responses are key, and therefore performance evaluation will focus on these two aspects using BYOD-RV on real devices.
- Envisaged setup: Device vendors are responsible to apply a minimal Android patch - update the system image with the library-injecting process and associated SELinux re-configuration. libbyod.so and RVMonitor.dex are dynamically placed on device by IT administration (e.g. through a work policy app or work SD card). Non-compliant devices must be flagged.