Infer

A static analyzer for catching bugs before you ship

Jules Villard
jul@fb.com

Facebook London
github.com/facebook/infer/
Programming is Hard

- Need to think of ALL possible cases
- Keep track of all possible values
- If it can be null, it will be null!
- Shipping bugs has consequences
  - Eg, users need to upgrade to get the fix
Code Quality

- Coding Good Practices: Tests, Code architecture, More Tests...
- Language Support: Null values? Try-with-resources? Type system?
  - Cannot always choose your language (legacy code, mobile apps, ...)
Static Analysis/Program Analysis

- Additional signal to developers
- Check all program paths and values
  - complement testing
- Palliative for tricky language features
  - complement compilers/type systems
Infer is a static analyzer written in OCaml for:

- Java
- C, C++, Objective-C

With the characteristics of being:

- Inter-procedural
- Incremental
A tool to detect bugs in Android and iOS apps before they ship

Facebook Infer is a static analysis tool - if you give Infer some Objective-C, Java, or C code, it produces a list of potential bugs. Anyone can use Infer to intercept critical bugs before they have shipped to people's phones, and help prevent crashes or poor performance.
The Infer "Hello World" Java example.

* Click the "Analyze" button to run Infer.
* Learn more about Infer at http://fbinfer.com

```java
public class Hello {
    void doesNotCauseNPE() {
        Pointers & a = Pointers.movReturnNull(10);
    }
}
```

This will display the output.
Demo
Infer Bug Types for C/C++

- Null Dereference
- Memory Leak
- Resource Leak
- Empty Vector Access [C++ only]
- Static Initialization Order Fiasco (using -a checker) [C++ only]
- Premature nil-Termination Argument
- ...

Infer Bug Types for Objective-C

- Null Dereference
- Memory Leak
- Resource Leak
- Retain Cycle
- Ivar not null checked
- Parameter not null checked
- ...

...
Infer Bug Types for Java

- Null Dereference
- Resource Leak
- Taint Analysis (with -a *quandary*)
- Performance Critical Calls Expensive Method (with -a *checker*)
- ...
Infer Bug Types for Android

- Context Leak
- Fragment Retains View (with `-a checker`)
In the Wild:
DuckDuckGo
DuckDuckGo’s bug report

Resource Leak with Cursor
RESOURCE LEAK: resource acquired to `c` by call to `query(...)` at line 329 is not released after line 336.
DuckDuckGo’s bug report

Null Dereference
NULL_DEREFERENCE: object feedObject last assigned on line 866 could be null and is dereferenced by call to feedItemSelected(...) at line 867
NULL_DEREFERENCE: object feedObject last assigned on line 866 could be null and is dereferenced by call to feedItemSelected(...) at line 867.
NULL_DEREFERENCE: object feedObject last assigned on line 866 could be null and is dereferenced by call to feedItemSelected(...) at line 867
NULL_DEREFERENCE: object feedObject last assigned on line 866 could be null and is dereferenced by call to feedItemSelected(...) at line 867
How does Infer work?
Infer Architecture

Source Code

+ Build System

SIL

Frontend

Analysis

Specs

Report

Languages:
Java
C
C++
ObjC

Build Systems:
ant
buck
cmake
gradle
maven
make
xcodebuild
Capture: Intermediate Language

```java
public class CodeSample {
    public String computeSomething(boolean flag) {
        if (flag) {
            return null;
        }
        else {
            return "something";
        }
    }

    public int doStuff() {
        String s = computeSomething(true);
        return s.length();
    }
}
```
Let’s focus on the “computeSomething” method
Capture: Intermediate Language

- Infer generate its Control Flow Graph (CFG)

```java
public class CodeSample {
    public String computeSomething(boolean flag) {
        if (flag) {
            return null;
        } else {
            return "something";
        }
    }

    public int doStuff() {
        String s = computeSomething(true);
        return s.length();
    }
}
```
Analysis: Pre- and Post-Conditions

- The way Infer expresses the possible states of the program

```
public String computeSomething(boolean flag) {
    if (flag) {
        return null;
    } else {
        return "something";
    }
}
```

This is called **PRE**condition

This is called **POST**condition

State before: flag = false

State after: flag = true

return "something"

return null
Analysis: Pre- and Post-Conditions

Infer finds two specifications

- **Precondition**
  - flag = true
- **Postcondition**
  - return = null

- **Precondition**
  - flag = false
- **Postcondition**
  - return = “something”
Analysis: Interprocedural

- Let's now focus on the “doStuff” method

```java
class CodeSample {
    public String computeSomething(boolean flag) {
        if (flag) {
            return null;
        } else {
            return "something";
        }
    }

    public int doStuff() {
        String s = computeSomething(true);
        return s.length();
    }
}
```

### Precondition
- **flag = false**
- **flag = true**

### Postcondition
- **return = "something"**
- **return = null**

Object returned by `computeSomething(true)` could be null and is dereferenced at line 13.
Another Analysis for Java: Eradicate

- Run with `-a eradicate`
- Checks that the code is *consistently* annotated with *@Nullable*
- Values not marked *@Nullable* are assumed non-null
- Guarantees absence of runtime NPE
Another Analysis for C/C++/ObjC: Linters

- Run with `-a linters`
- AST-based, syntactic checks
- Add your own checks using the DSL: `infer --linters-def-file ./linters.al ...

```plaintext
// a property with a pointer type should not be declared `assign`
DEFINE-CHECKER ASSIGN_POINTER_WARNING = {
  SET report_when = WHEN is_assign_property()
      AND is_property_pointer_type()
      HOLDS-IN-NODE ObjCPropertyDecl;
  SET message = ...; SET suggestion = ...;
};
```

`linters.al`
Deploying Infer
Slow
Deployment Model
Nightly, Bug List

Users
Faster Deployment Model
CI system

Phabricator

Code reviewers

Developer

Product

CI system

Performance tests
Continuous UI correctness tests
This file was added.

```java
public class CodeSample {
    public String computeSomething(boolean flag) {
        if (flag) {
            return null;
        }
        else {
            return "something";
        }
    }

    public int doStuff() {
        String s = computeSomething(true);
        return s.length();
    }
}
```

There may be a Null Dereference: object s last assigned on line 12 could be null and is dereferenced at line 13
CI system

Phabricator

Code reviewers

Developer

Product

Performance tests
Continuous UI correctness tests

CI system
Diff Analysis

1. Run infer on top revision → report-top.json
2. Run infer on base revision → report-base.json
4. Report new issues only

Upcoming support for this workflow in infer itself
Current status:
In a typical month...

- Infer runs on thousands of modifications to Facebook's mobile code bases
- Hundreds of potential bugs are reported by Infer and fixed by FB developers. (Fix rate: 70% approx in recent months)
Infer

A static analyzer for catching bugs before you ship

Jules Villard
jul@fb.com

Facebook London

github.com/facebook/infer/