# Infer: Deploying Static Analysis at Facebook

Cristiano Calcagno

Facebook

Software Correctness And Reliability 2015

joint work with: Peter O'Hearn, Dino Distefano, Sam Blackshear, Jeremy Dubreil, Dominik Gabi, Pieter Hooimeijer, Andrzej Kotulski, Martino Luca, Irene Papakostantinou, Jim Purbrick, Dulma Rodriguez, Jules Villard

MOVE FAST AND BREAK THINGS

facebook

facebook.com/careers

THE FOOLISH WAIT

facebook

facebook.com/careers

DONE IS BETTER THAN PERFECT

facebook

facebook.com/careers

# Scalable Shape Analysis for Systems Code, CAV'08

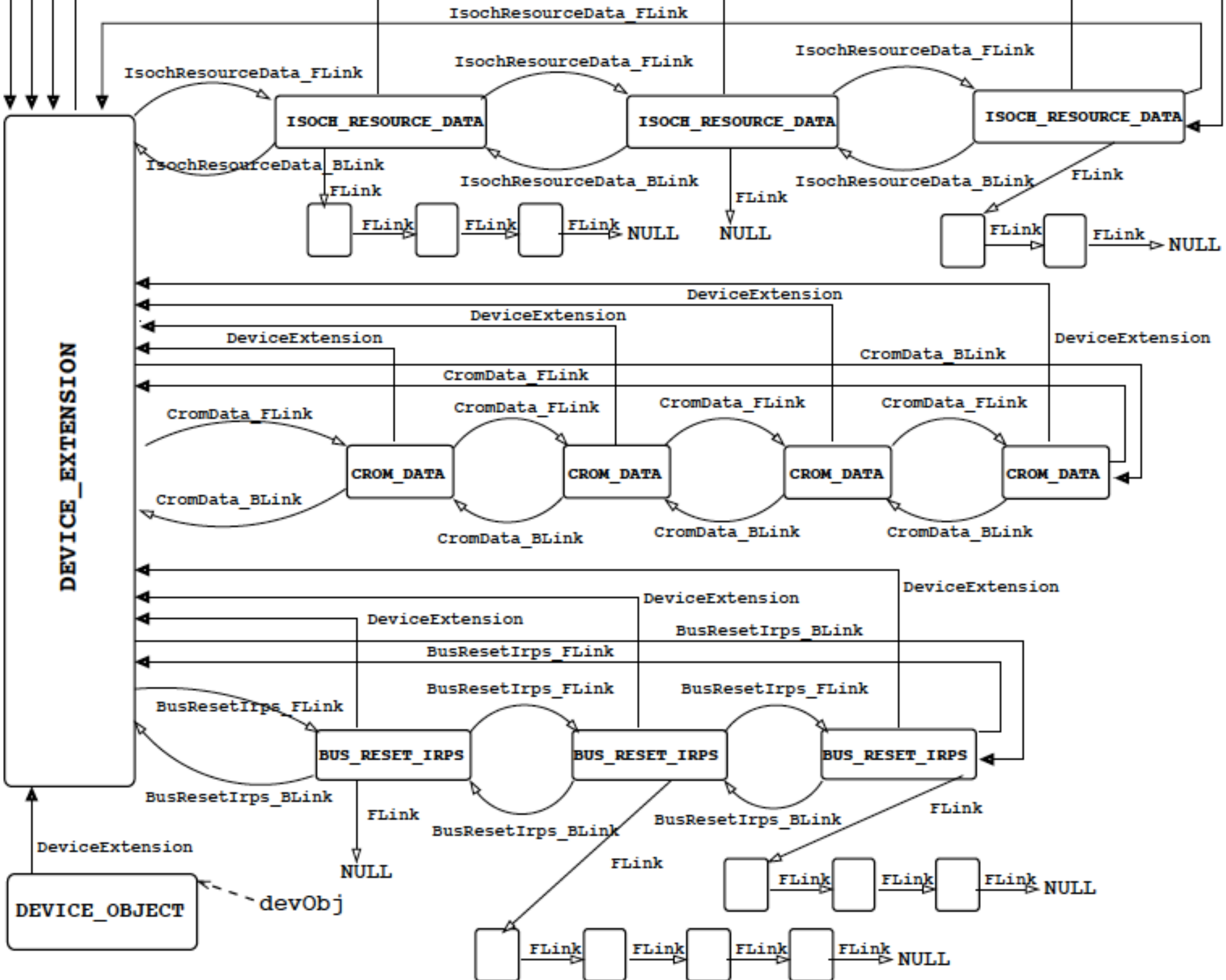
Yang, Lee, Berdine, Calcagno, Cook, Distefano, O'Hearn

## *Proofs for Device Drivers*

| Program | LOC | Sec | Mb |
|---|---|---|---|
| `pci-driver.c` | 2532 | 0.75 | 3.19 |
| `cdrom.c` | 6218 | 91.45 | 84.79 |
| `t1394Diag.c` | 10240 | 137.78 | 73.24 |
| `md.c` | 6635 | 1819.53 | 1010.81 |
| `ll_rw_blk.c` | 5469 | 947.20 | 511.43 |

$>$ 60 bugs found in these drivers. We inserted our own fixes. Then..

Pointer Safety proofs found for fixed drivers


DEVICE_EXTENSION
DEVICE_OBJECT
devObj
IsochResourceData_FLink
IsochResourceData_BLink
ISOCH_RESOURCE_DATA
FLink
NULL
DeviceExtension
CromData_FLink
CromData_BLink
CROM_DATA
BusResetIrps_FLink
BusResetIrps_BLink
BUS_RESET_IRPS

# Compositionality

The meaning of a composite expression is determined by the meanings of its constituent parts.

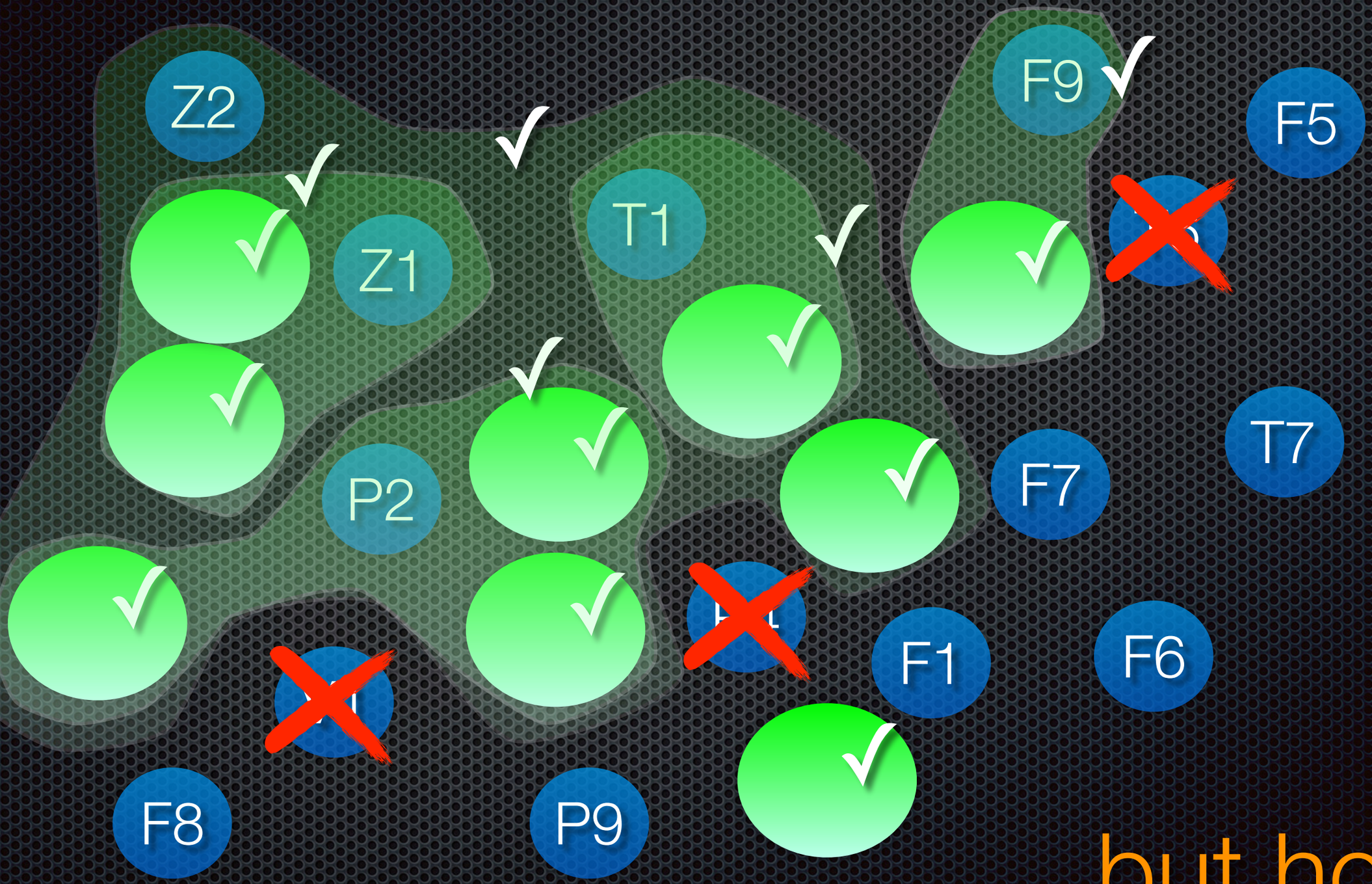
Frege's Principle

Gottlob Frege
(1848-1925)

In program analysis:

The analysis result of a composite program is computed from the analysis results of its parts

# Compositional approach




...but how?

# Abduction

## Abductive Inference (Charles Peirce, circa 1900, writing about the scientific process)

"Abduction is the process of forming an explanatory hypothesis.
It is the only logical operation which introduces any new idea"

The surprising fact, C, is observed.
But if A were true, C would be a matter of course.
Hence, there is reason to suspect that A is true.

Charles Pierce (Pragmatism and Abduction)

# *The Abduction Question*

$$x \mapsto \texttt{nil} * ? \;\vdash\; \mathit{list}(x) * \mathit{list}(y)$$

# *The Abduction Question*

$$x \mapsto \texttt{nil} * \mathit{list}(y) \;\vdash\; \mathit{list}(x) * \mathit{list}(y)$$

# *The Abduction Question*

$$x \mapsto - * ?? \;\vdash\; y \mapsto - * \texttt{true}$$

# *The Abduction Question*

▶

$$x \mapsto - * \left(x = y \wedge \mathtt{emp}\right) \;\vdash\; y \mapsto - * \mathtt{true}$$

# *The Abduction Question*

▶

$$x \mapsto - * y \mapsto - \;\vdash\; y \mapsto - * \texttt{true}$$

# *Abduction Example: Inferring a pre/post pair*

```
1 void p(list-item *y) {
2     list-item *x;
3     x=malloc(sizeof(list-item));
4     x⟶tail = 0;
5     merge(x,y);
6     return(x); }
```

Abductive Inference:

Given Summary/spec: $\{\mathsf{list}(x) * \mathsf{list}(y)\}\mathit{merge}(x, y)\{\mathsf{list}(x)\}$

# *Abduction Example: Inferring a pre/post pair*

```
1 void p(list-item *y) {                    emp
2     list-item *x;
3     x=malloc(sizeof(list-item));
4     x⟶tail = 0;
5     merge(x,y);
6     return(x); }
```

Abductive Inference:

Given Summary/spec: $\{\mathsf{list}(x) * \mathsf{list}(y)\}\mathit{merge}(x, y)\{\mathsf{list}(x)\}$

# *Abduction Example: Inferring a pre/post pair*

```
1 void p(list-item *y) {
2     list-item *x;
3     x=malloc(sizeof(list-item));
4     x⟶tail = 0;
5     merge(x,y);
6     return(x); }
```

emp

$x \mapsto 0$

Abductive Inference:

Given Summary/spec: $\{\mathsf{list}(x) * \mathsf{list}(y)\}\mathit{merge}(x, y)\{\mathsf{list}(x)\}$

# *Abduction Example: Inferring a pre/post pair*

```
1 void p(list-item *y) {
2     list-item *x;
3     x=malloc(sizeof(list-item));
4     x⟶tail = 0;
5     merge(x,y);
6     return(x); }
```

emp

$x \mapsto 0$

Abductive Inference: $x \mapsto 0 * ?$ $\vdash \mathsf{list}(x) * \mathsf{list}(y)$

Given Summary/spec: $\{\mathsf{list}(x) * \mathsf{list}(y)\}\mathit{merge}(x, y)\{\mathsf{list}(x)\}$

# *Abduction Example: Inferring a pre/post pair*

```
1 void p(list-item *y) {                    emp
2     list-item *x;
3     x=malloc(sizeof(list-item));
4     x⟶tail = 0;                          x ↦ 0
5     merge(x,y);
6     return(x); }
```

Abductive Inference: $x \mapsto 0 * \ \mathsf{list}(y) \ \vdash \mathsf{list}(x) * \mathsf{list}(y)$

Given Summary/spec: $\{\mathsf{list}(x) * \mathsf{list}(y)\} merge(x, y) \{\mathsf{list}(x)\}$

# *Abduction Example: Inferring a pre/post pair*

```
1 void p(list-item *y) {
2     list-item *x;
3     x=malloc(sizeof(list-item));
4     x⟶tail = 0;
5     merge(x,y);
6     return(x); }
```

emp

list($y$)

$x \mapsto 0$

Abductive Inference: $x \mapsto 0 * \text{list}(y) \vdash \text{list}(x) * \text{list}(y)$

Given Summary/spec: $\{\text{list}(x) * \text{list}(y)\}\mathit{merge}(x, y)\{\text{list}(x)\}$

# *Abduction Example: Inferring a pre/post pair*

| | | |
|---|---|---|
| 1 void p(list-item *y) { | emp | list($y$) |
| 2 list-item *x; | | |
| 3 x=malloc(sizeof(list-item)); | | |
| 4 x$\rightarrow$tail $= 0$; | $x \mapsto 0$ | |
| 5 merge(x,y); | list($x$) | |
| 6 return(x); } | | |

Abductive Inference: $x \mapsto 0 * \text{list}(y) \vdash \text{list}(x) * \text{list}(y)$

Given Summary/spec: $\{\text{list}(x) * \text{list}(y)\} merge(x, y) \{\text{list}(x)\}$

# *Abduction Example: Inferring a pre/post pair*

```
1 void p(list-item *y) {                    emp          list(y)
2     list-item *x;
3     x=malloc(sizeof(list-item));
4     x⟶tail = 0;                          x ↦ 0
5     merge(x,y);                          list(x)
6     return(x); }                                      list(ret)
```

Abductive Inference: $x \mapsto 0 * \mathsf{list}(y) \vdash \mathsf{list}(x) * \mathsf{list}(y)$

Given Summary/spec: $\{\mathsf{list}(x) * \mathsf{list}(y)\}\mathit{merge}(x, y)\{\mathsf{list}(x)\}$

# *Abduction Example: Inferring a pre/post pair*

| | | | |
|---|---|---|---|
| 1 void p(list-item *y) { | emp | list($y$) | (Inferred Pre) |
| 2 list-item *x; | | | |
| 3 x=malloc(sizeof(list-item)); | | | |
| 4 x$\rightarrow$tail $= 0$; | $x \mapsto 0$ | | |
| 5 merge(x,y); | list($x$) | | |
| 6 return(x); } | | list($ret$) | (Inferred Post) |

Abductive Inference: $x \mapsto 0 * \text{list}(y) \vdash \text{list}(x) * \text{list}(y)$

Given Summary/spec: $\{\text{list}(x) * \text{list}(y)\} merge(x, y) \{\text{list}(x)\}$

# Bi-Abduction

Synthesising both missing resources (anti-frame) and unneeded resources (frame) gives rise to a new notion

Bi-Abduction:

given A and B compute ?antiframe and ?frame such that

$$A * ?antiframe \vdash B * ?frame$$

# Bi-Abductive symbolic execution

Pre: list(x) * list(y)

void foo(list_item *x,list_item *y)

Post: list(x)

```
node* p(list_item *y) {
  node *x, *z;
1   x=malloc(sizeof(list_item)); x->tail =  0;
2   z=malloc(sizeof(list_item)); z->tail =  0;
3   foo(x,y);
4   foo(x,z);
5   return x;
  }
```

list(y)

emp

$x \mapsto 0$

$x \mapsto 0 * z \mapsto 0$

$\mathsf{list}(x) * z \mapsto 0$

$\mathsf{list}(x)$

$\mathsf{list}(\mathtt{ret})$

## Bi-abductive prover

POPL'09/JACM'11

Compositional Shape Analysis by means of Bi-Abduction

| Program | KLOC | Num. Procs | Proven Procs | Proven % | Time |
|---|---|---|---|---|---|
| Linux kernel 2.6.30 | 3032 | 143768 | 86268 | 60.0 | 9617.44 |
| Gimp 2.4.6 | 705 | 16087 | 8624 | 53.6 | 8422.03 |
| Gtk 2.18.9 | 511 | 18084 | 9657 | 53.4 | 5242.23 |
| Emacs 23.2 | 252 | 3800 | 1630 | 42.9 | 1802.24 |
| Glib 2.24.0 | 236 | 6293 | 3020 | 48.0 | 3240.81 |
| Cyrus imapd 2.3.13 | 225 | 1654 | 1150 | 68.2 | 1131.72 |

# Facebook Engineering

Computers/Technology

Liked

Timeline | About | Photos | Likes | More

separation logic  abstract interpretation  bi-abduction

facebook

$$\frac{\{I\}\ F\ \{E\} \qquad N \vdash R}{\{\mathbf{I} * \mathbf{N}\}\ \mathbf{F}\ \{\mathbf{E} * \mathbf{R}\}}$$



...

# Where to start?

# Mobile first

# How to deploy?

# Errors on Mobile code

- Focus on crucial categories of bugs (affected users)
- Null Pointer Exceptions
- Resource Leaks (streams, output streams, readers, writers, sockets, http connections, cursors, and json parsers)
- Memory Leaks
- Retain cycles

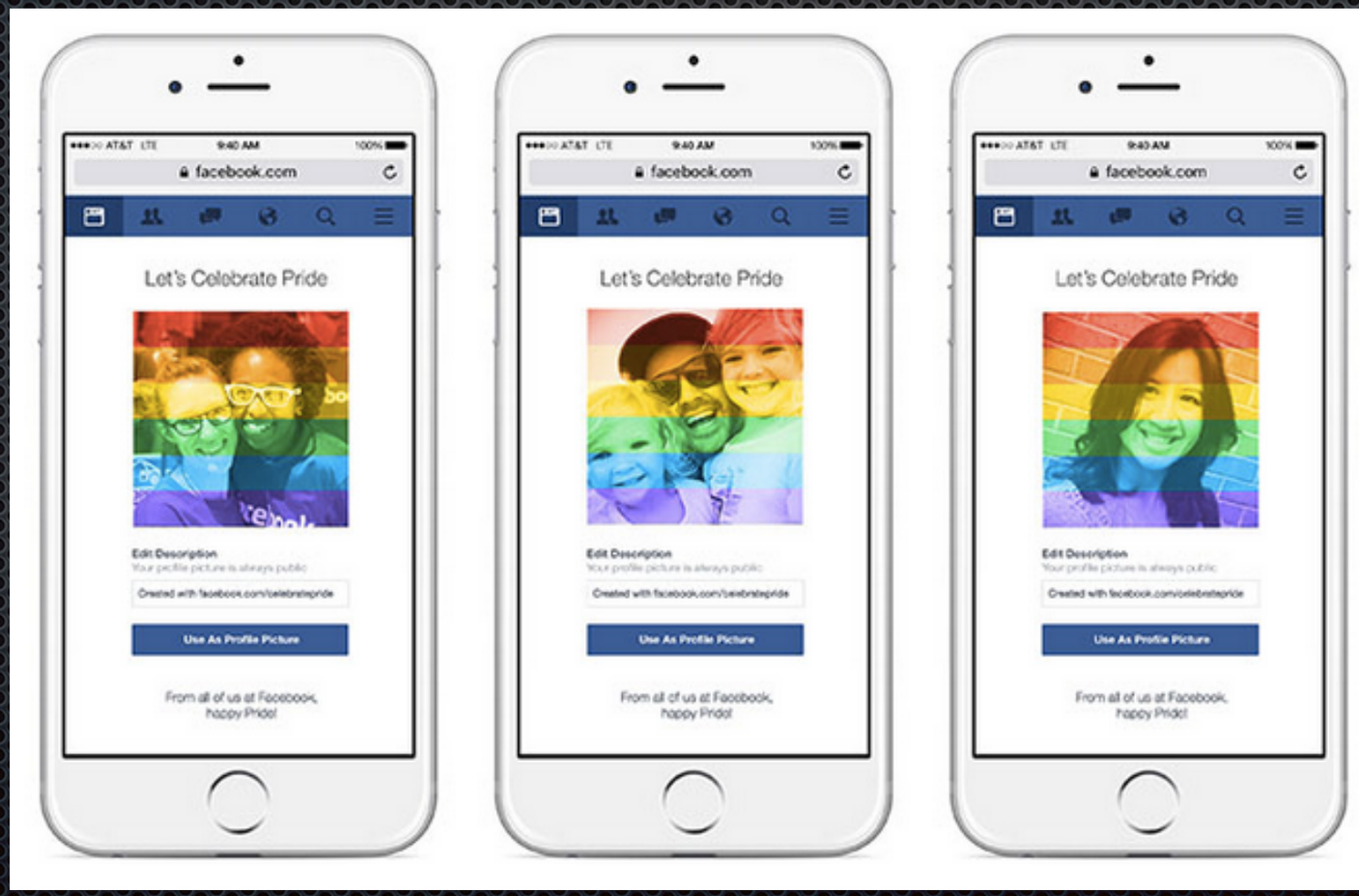
# Slow
# Deployment Model

Nightly, Bug List

# Moving Fast

72 hours to launch Celebrate Pride

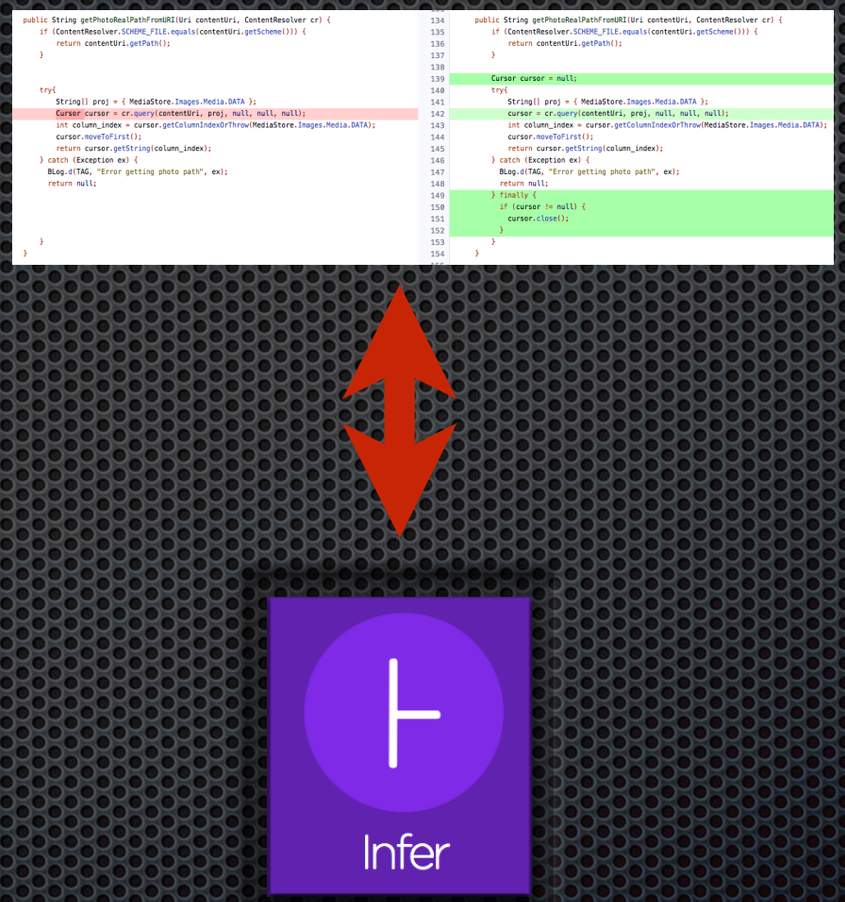
Omid Aziz Derrick McMillen Tony Liu



# Perpetual Development

We don't want computers waiting on humans

# Faster Deployment Model

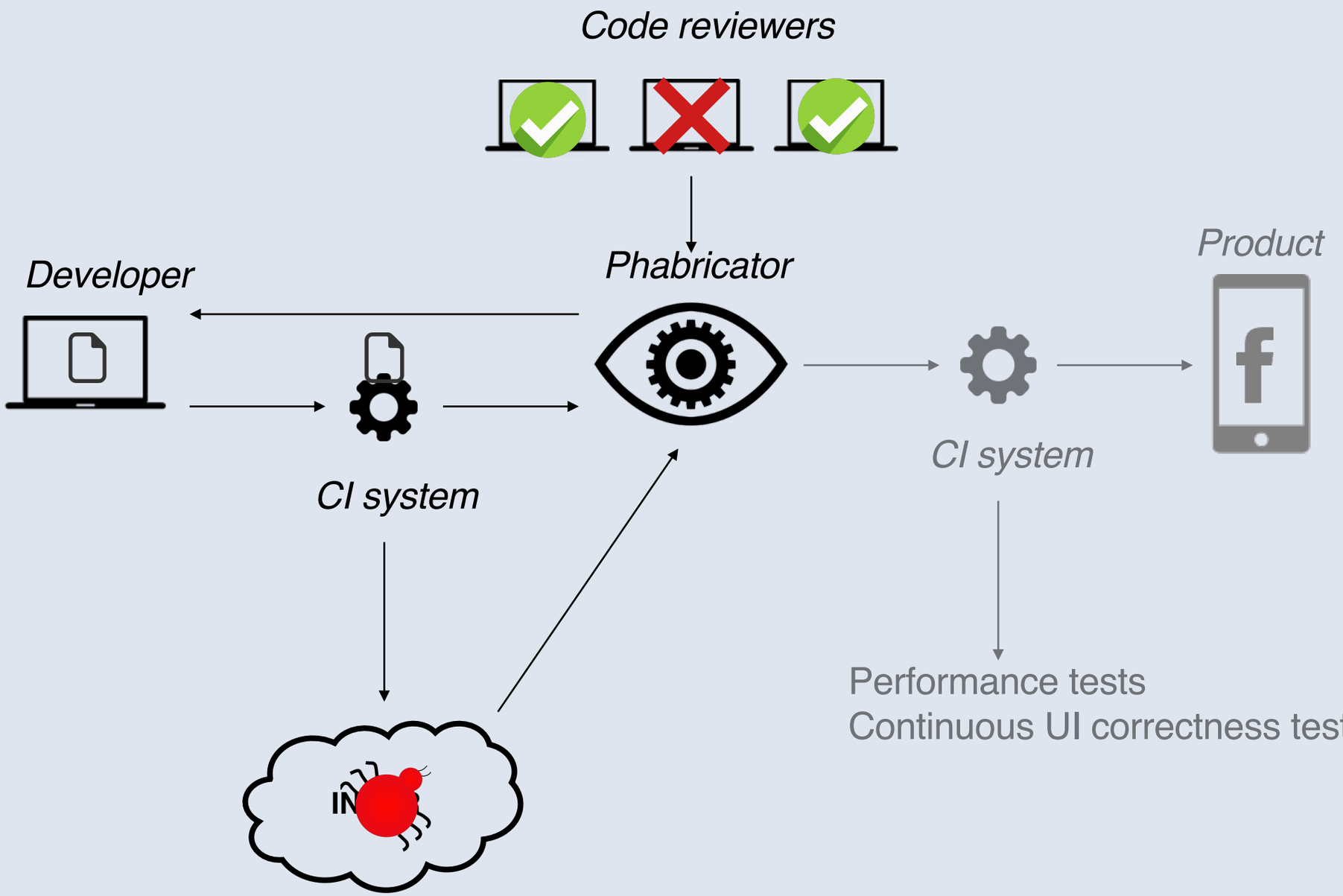


```
public String getPhotoRealPathFromURI(Uri contentUri, ContentResolver cr) {
    if (ContentResolver.SCHEME_FILE.equals(contentUri.getScheme())) {
        return contentUri.getPath();
    }

    try{
        String[] proj = { MediaStore.Images.Media.DATA };
        Cursor cursor = cr.query(contentUri, proj, null, null, null);
        int column_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
        cursor.moveToFirst();
        return cursor.getString(column_index);
    } catch (Exception ex) {
      BLog.d(TAG, "Error getting photo path", ex);
      return null;


    }
}
```

```
134     public String getPhotoRealPathFromURI(Uri contentUri, ContentResolver cr) {
135         if (ContentResolver.SCHEME_FILE.equals(contentUri.getScheme())) {
136             return contentUri.getPath();
137         }
138
139         Cursor cursor = null;
140         try{
141             String[] proj = { MediaStore.Images.Media.DATA };
142             cursor = cr.query(contentUri, proj, null, null, null);
143             int column_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
144             cursor.moveToFirst();
145             return cursor.getString(column_index);
146         } catch (Exception ex) {
147           BLog.d(TAG, "Error getting photo path", ex);
148           return null;
149         } finally {
150           if (cursor != null) {
151             cursor.close();
152           }
153         }
154     }
```


Code reviewers
Developer
Phabricator
Product
CI system
CI system
Performance tests
Continuous UI correctness tests

# Comments

**infer_report_example/CodeSample.java** View Options

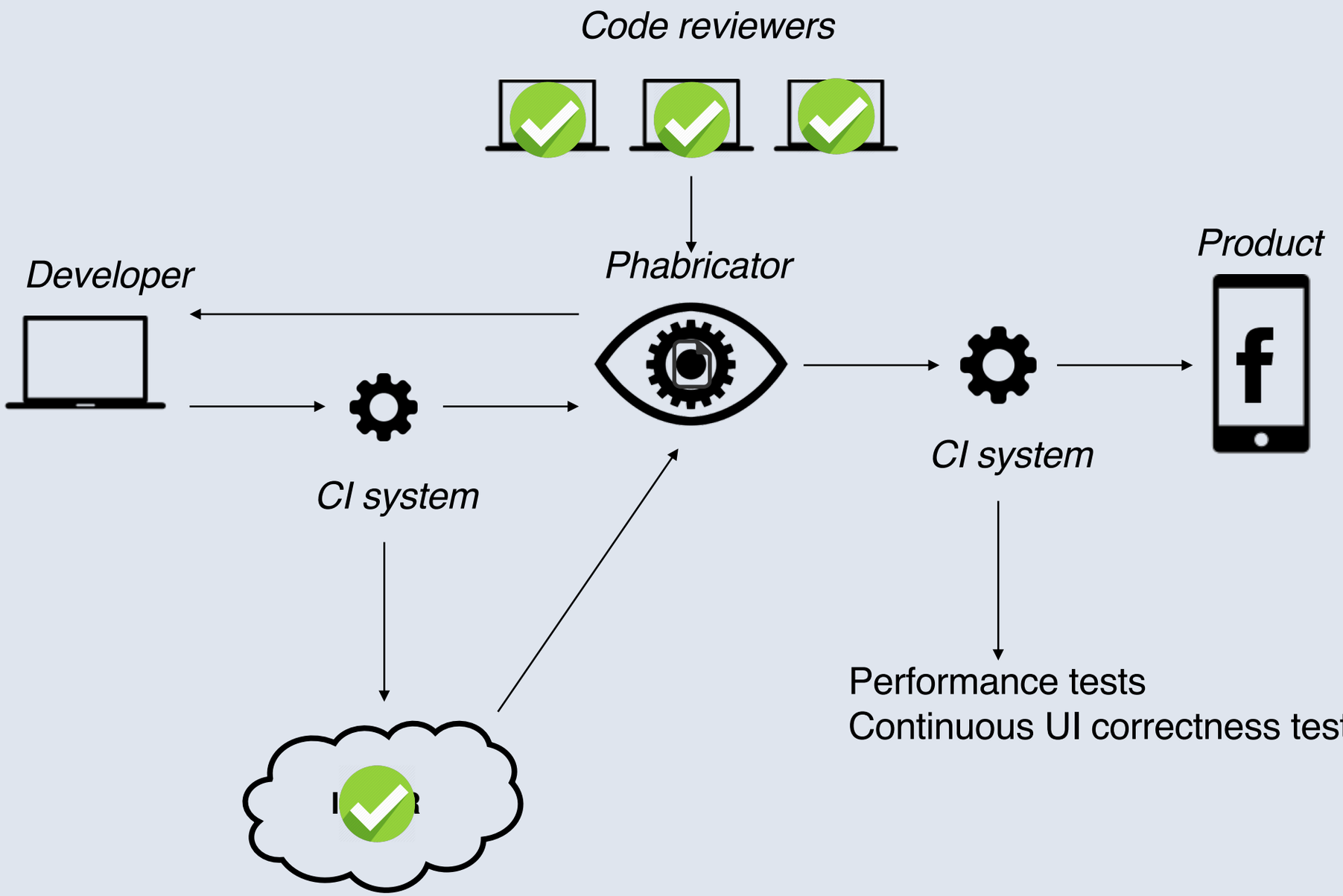
This file was **added**.

```
1  public class CodeSample {
2    public String computeSomething(boolean flag) {
3      if (flag) {
4        return null;
5      }
6      else {
7        return "something";
8      }
9    }
10
11   public int doStuff() {
12     String s = computeSomething(true);
13     return s.length();
```

*Line 13* Previous · Next · Reply

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

```
14   }
15 }
```


Code reviewers
Developer
Phabricator
Product
CI system
CI system
Performance tests
Continuous UI correctness tests

# Software Verification in the Perpetual Development Era

- Scientific and Social challenge
  - Full automation and integration with development environment
  - Scalability
  - Precision
  - Fast Reporting, 10min on code diffs

SAVE DEVELOPER TIME

# Moving Fast

Just wanted to shout out: not sure, what was changed, but infer is now reporting so amazingly fast on fbandroid. way better experience. Previously I was completely loosing context on the task and switched to other ones as results would come in half an hour. Now they're almost instant as I re-read the diff and changed the workflow. Thanks, guys!

Like Comment Share

You, Cristiano Calcagno, [redacted] like this.

# Breaking Things

[redacted] 4/10, 9:49am

hi - Apologies for not addressing it earlier. I just commented, that it's a false positive because there is a check for it upstream.

**Peter O'Hearn** 4/10, 9:58am

Ah, that is pretty subtle. Thanks a lot for pointing that out!

[redacted] 4/10, 9:59am

Thanks for following up!

i have found it to be useful saved me trouble ahead instead of seeing NPEs when testing

Thanks for the tool!

```
18  int main(int argc, const char *argv[]) {
19    C *c = [[C alloc] init];
20    c.handler = ^(NSString *name) {
21      c.name = name;
22    };
23    return 0;
```

**trunkagent** *Line 23* Previous · Next · Reply

There may be a Retain Cycle: Retain cycle involving the following objects: (1) an object of class C retaining another object via instance variable _handler, (2) a block capturing c; at line 23, column 3 (error trace: TV7012587).

```
24  }
```

# K-9 Android email client

Report: Resource Leak: resource acquired by call to FileWriter(...) at line 143 is not released after line 147.

```
143 -           FileWriter fstream = new FileWriter(OUTPUT_FILE);
144 -           BufferedWriter out = new BufferedWriter(fstream);
145 -           out.write(content);
146 -           out.close();
```

# OpenSSL examples

To be honest, there's not much to tell. Almost nothing strange was found. Those errors described in that earlier article are fixed by now. OpenSSL is a quality project; the library has been already checked by many tools (Clang, Cppcheck, Coverity, DoubleCheck, Coccinelle, Klocwork, etc.). So, the library is cleaned out, it would be a feat to find even one error there.

PVS Studio Blog

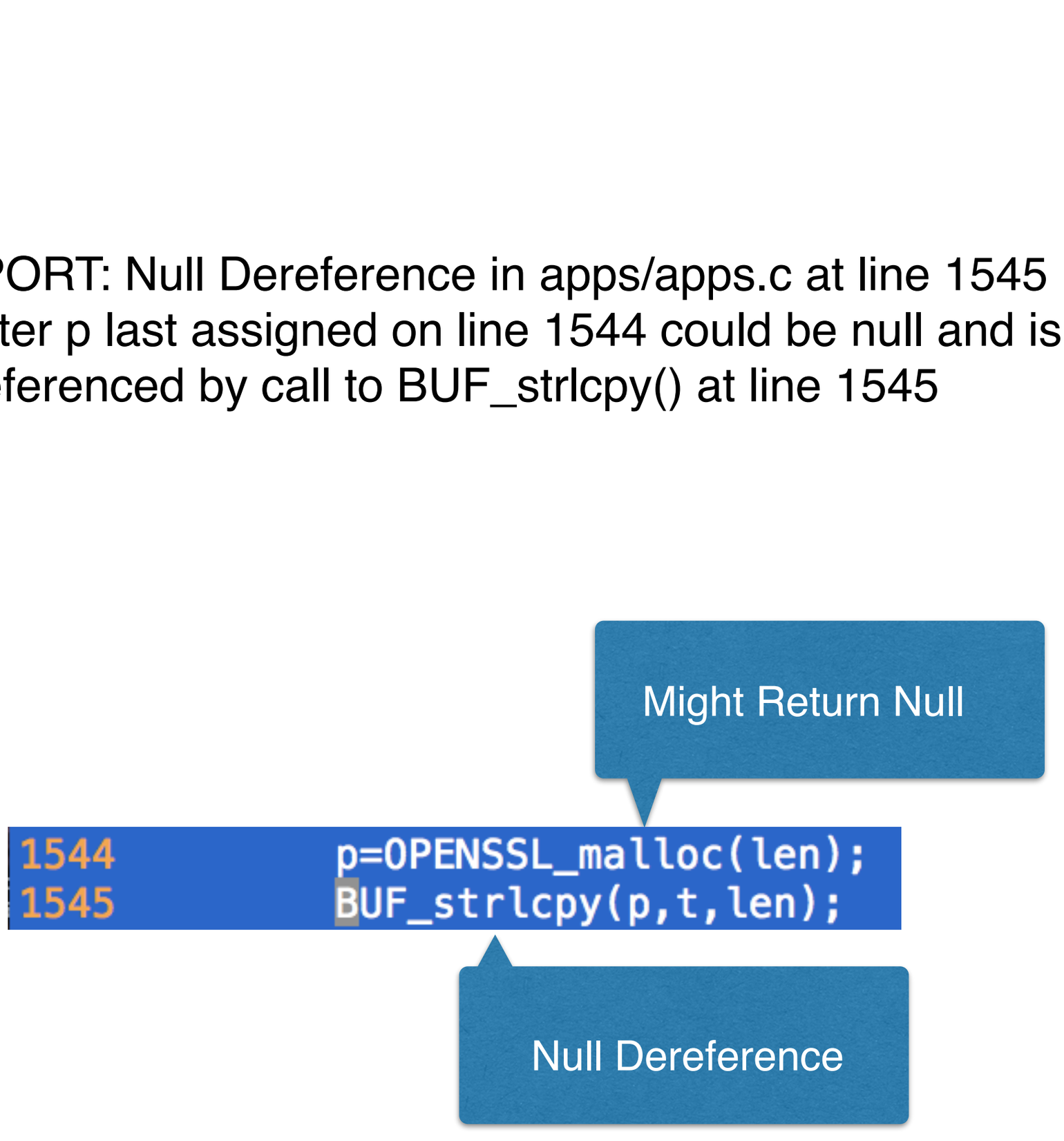
**Jim Purbrick** feeling proud
4 July ·

Facebook London finding bugs in critical open-source software.

**#3403: Null dereference and memory leak reports for openssl-1.0.1h from Facebook's Infer static...**
RT.OPENSSL.ORG

Like · Comment · Share

REPORT: Null Dereference in apps/apps.c at line 1545 pointer p last assigned on line 1544 could be null and is dereferenced by call to BUF_strlcpy() at line 1545



```
368 #define OPENSSL_malloc(num)     CRYPTO_malloc((int)num,__FILE__,__LINE__)
```

```
void *CRYPTO_malloc(int num, const char *file, int line)
    {
    void *ret = NULL;

    if (num <= 0) return NULL;

    allow_customize = 0;
    if (malloc_debug_func != NULL)
        {
        allow_customize_debug = 0;
        malloc_debug_func(NULL, num, file, line, 0);
        }
    ret = malloc_ex_func(num,file,line);
#ifdef LEVITTE_DEBUG_MEM
    fprintf(stderr, "LEVITTE_DEBUG_MEM:         > 0x%p (%d)\n", ret, num);
#endif
    if (malloc_debug_func != NULL)
        malloc_debug_func(ret, num, file, line, 1);

#ifndef OPENSSL_CPUID_OBJ
        /* Create a dependency on the value of 'cleanse_ctr' so our memory
         * sanitisation function can't be optimised out. NB: We only do
         * this for >2Kb so the overhead doesn't bother us. */
        if(ret && (num > 2048))
    {   extern unsigned char cleanse_ctr;
                ((unsigned char *)ret)[0] = cleanse_ctr;
    }
#endif

    return ret;
    }
```

```
100 size_t BUF_strlcpy(char *dst, const char *src, size_t size
101             {
102             size_t l = 0;
103             for(; size > 1 && *src; size--)
104                     {
105                     *dst++ = *src++;
106                     l++;
107                     }
108             if (size)
109                     *dst = '\0';
110             return l + strlen(src);
111             }
```

# In between true and false bugs

Might Return Null

```
2777
2778        revtm = X509_gmtime_adj(NULL, 0);
2779
2780        i = revtm->length + 1;
```

Null Dereference

REMARKS:
— The definition of X509_gmtime_adj is in crypto/x509/x509_vfy.c. It calls  X509_time_ad which calls X509_time_adj_ex which calls several other things which can return NULL.
— The conditions under which X509_gmtime_adj(NULL, 0) returns null are somewhat complex. Not verified that these conditions will arise.
— Calls to X509_gmtime_adj(NULL, 0) are checked for NULL before dereference elsewhere in the codebase; for example, in crypto/cms/cms_sd.c at line 471.

```c
ASN1_TIME *X509_gmtime_adj(ASN1_TIME *s, long adj)
{
    return X509_time_adj(s, adj, NULL);
}

ASN1_TIME *X509_time_adj(ASN1_TIME *s, long adj, time_t *in_tm)
    {
    time_t t;

    if (in_tm) t = *in_tm;
    else time(&t);

    t+=adj;
    if (!s) return ASN1_TIME_set(s, t);
    if (s->type == V_ASN1_UTCTIME) return ASN1_UTCTIME_set(s,t);
    return ASN1_GENERALIZEDTIME_set(s, t);
    }
```

```c
ASN1_TIME *ASN1_TIME_set(ASN1_TIME *s, time_t t)
    {
    return ASN1_TIME_adj(s, t, 0, 0);
    }

ASN1_TIME *ASN1_TIME_adj(ASN1_TIME *s, time_t t,
                int offset_day, long offset_sec)
    {
    struct tm *ts;
    struct tm data;

    ts=OPENSSL_gmtime(&t,&data);
    if (ts == NULL)
        {
        ASN1err(ASN1_F_ASN1_TIME_ADJ, ASN1_R_ERROR_GETTING_TIME);
        return NULL;
        }
    if (offset_day || offset_sec)
        {
        if (!OPENSSL_gmtime_adj(ts, offset_day, offset_sec))
            return NULL;
        }
    if((ts->tm_year >= 50) && (ts->tm_year < 150))
            return ASN1_UTCTIME_adj(s, t, offset_day, offset_sec);
    return ASN1_GENERALIZEDTIME_adj(s, t, offset_day, offset_sec);
    }
```

# Limitations, I

- Bugs reported in Java
  - Resource leak
  - Null dereference
- Bugs reported in C and Objective-C
  - Resource leak
  - Memory leak
  - Null dereference
  - Parameter not null checked
  - Ivar not null checked
  - Premature nil termination argument
- Bugs reported only in Objective-C
  - Retain cycle

# Limitations, II

A different dimension in which Infer is limited concerns language features. Infer either does not understand or has a weak treatment of

- Concurrency, including Java's Concurrency Utilities and iOS's Grand Central Dispatch
- Dynamic dispatch
- Reflection
- Android lifecycles
- Arithmetic
- and more

# Limitations, III

[redacted] Fun question: can infer spot the bug for us so we don't let it happen in the future?

cc Peter O'Hearn

I wonder if Infer could have found this crasher. Peter?

We discussed this SEV in review today: [redacted]

Infer came up as a possible way to catch this. Have we worked with the [redacted] team yet? Cristiano are you in MPK still? Now might be a good time to hop in 🙂

# 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: 80% approx in recent months)
- Millions of calls are issued to a bi-abductive theorem prover for Separation Logic

$$A * ?antiframe \vdash B * ?frame$$

# Facebook Open-Sources Infer To Help Developers Identify Bugs Before They're Shipped

Posted Jun 11, 2015 by **Frederic Lardinois** (@fredericl)

2,325 SHARES

Next Story

COMMENT

aristus 42 days ago

Legend has it there is a small room at FBHQ, containing a quorum of OCaml committers, all of them French for some reason, hacking away at level of abstraction beyond the ken of mortal man.

theblatte 42 days ago

I'm posting from that very room right now.

**Craig Marvelley** @craigmarvelley · Jun 24

Just fixed a bunch of null dereference errors that Facebook's Infer tool flagged up. fbinfer.com

**Rob Smedley** @rrsmedley · 12h

My initial thoughts after about an hour of playing is the this is pretty bad ass

leaks in my project using:
infer -- gradle build

THIS JOURNEY

1% FINISHED

Don't (only) be whole program

Be

Compositional