Infer: Deploying Static Analysis at Facebook

Cristiano Calcagno

Facebook

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

Software Correctness And Reliability 2015
MOVE FAST AND BREAK THINGS

THE FOOLISH WAIT

DONE IS BETTER THAN PERFECT
Proofs for Device Drivers

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Sec</th>
<th>Mb</th>
</tr>
</thead>
<tbody>
<tr>
<td>pci-driver.c</td>
<td>2532</td>
<td>0.75</td>
<td>3.19</td>
</tr>
<tr>
<td>cdrom.c</td>
<td>6218</td>
<td>91.45</td>
<td>84.79</td>
</tr>
<tr>
<td>t1394Diag.c</td>
<td>10240</td>
<td>137.78</td>
<td>73.24</td>
</tr>
<tr>
<td>md.c</td>
<td>6635</td>
<td>1819.53</td>
<td>1010.81</td>
</tr>
<tr>
<td>ll_rw_blk.c</td>
<td>5469</td>
<td>947.20</td>
<td>511.43</td>
</tr>
</tbody>
</table>

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

Pointer Safety proofs found for fixed drivers
Compositionality

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

Frege’s Principle

In program analysis:

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

Gottlob Frege (1848-1925)
Compositional approach

...but how?
Abduction

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.

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”

Charles Pierce (Pragmatism and Abduction)
The Abduction Question

\[ x \leftarrow \text{nil} \quad ? \quad \text{list}(x) \ast \text{list}(y) \]
The Abduction Question

\[ x \mapsto \text{nil} \ast \text{list}(y) \vdash \text{list}(x) \ast \text{list}(y) \]
The Abduction Question

\[ x \mapsto - * ?? \quad \vdash \quad y \mapsto - * true \]
The Abduction Question

\[ x \mapsto \neg \star (x = y \land \text{emp}) \quad \vdash \quad y \mapsto \neg \star \text{true} \]
The Abduction Question

\[ x \leftrightarrow \neg \ast y \leftrightarrow \neg \vdash y \leftrightarrow \neg \ast \text{true} \]
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: \{list(x) * list(y)\} merge(x, y)\{list(x)\}
Abduction Example: Inferring a pre/post pair

```c
void p(list-item *y) {
    list-item *x;
    x = malloc(sizeof(list-item));
    x->tail = 0;
    merge(x, y);
    return(x); }
```

Abductive Inference:

Given Summary/spec:  
\{ list(x) \land list(y) \} merge(x, y) \{ list(x) \}
Abduction Example: Inferring a pre/post pair

```c
void p(list-item *y) {
  list-item *x;
  x = malloc(sizeof(list-item));
  x->tail = 0;
  merge(x, y);
  return(x);
}
```

Abductive Inference:

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

Abductive Inference: \( x \leftarrow 0 \) * ? \( \vdash \text{list}(x) \ast \text{list}(y) \)

Given Summary/spec: \( \{\text{list}(x) \ast \text{list}(y)\} \text{merge}(x, y) \{\text{list}(x)\} \)
Abduction Example: Inferring a pre/post pair

void p(list-item *y) {
  list-item *x;
  x = malloc(sizeof(list-item));
  x->tail = 0;
  merge(x, y);
  return(x);
}

Abductive Inference:  
\[
\begin{align*}
g & \leftarrow 0 * \quad \text{list}(y) \quad \vdash \quad \text{list}(x) * \text{list}(y)
\end{align*}
\]

Given Summary/spec: 
\[
\{ \text{list}(x) * \text{list}(y) \} \quad \text{merge}(x, y) \{ \text{list}(x) \}\]
Abduction Example: Inferring a pre/post pair

```c
void p(list-item *y) {
    list-item *x;
    x = malloc(sizeof(list-item));
    x->tail = 0;
    merge(x, y);
    return(x);
}
```

Abductive Inference: \( x \mapsto 0 \uparrow \text{list}(y) \vdash \text{list}(x) \ast \text{list}(y) \)

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

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

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

Abductive Inference:

```
x \rightarrow 0 \quad \text{list(y)} \models \text{list(x)} \quad \text{list(ret)}
```

Given Summary/spec:

```
\{ \text{list(x)} \} \cdot \text{merge(x, y)} \{ \text{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) (Inferred Pre)

x \mapsto 0

list(x)

list(ret) (Inferred Post)

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

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

\[ A \times ?antiframe \vdash B \times ?frame \]
Bi-Abductive symbolic execution

```c
node* p(list_item *y) {
    node *x, *z;
    x = malloc(sizeof(list_item)); x->tail = 0;
    z = malloc(sizeof(list_item)); z->tail = 0;
    foo(x, y);
    foo(x, z);
    return x;
}
```

Pre: list(x) * list(y)
void foo(list_item *x, list_item *y)
Post: list(x)

Bi-abductive prover

```
list(x) * x ⇠ 0 list(z) * z ⇠ 0
list(x) * z ⇠ 0
```

```c
emp
x ⇠ 0
x ⇠ 0 * z ⇠ 0
list(x) * z ⇠ 0
list(x)
list(ret)
```
Compositional Shape Analysis by means of Bi-Abduction

<table>
<thead>
<tr>
<th>Program</th>
<th>KLOC</th>
<th>Num. Procs</th>
<th>Proven Procs</th>
<th>Proven %</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux kernel 2.6.30</td>
<td>3032</td>
<td>143768</td>
<td>86268</td>
<td>60.0</td>
<td>9617.44</td>
</tr>
<tr>
<td>Gimp 2.4.6</td>
<td>705</td>
<td>16087</td>
<td>8624</td>
<td>53.6</td>
<td>8422.03</td>
</tr>
<tr>
<td>Gtk 2.18.9</td>
<td>511</td>
<td>18084</td>
<td>9657</td>
<td>53.4</td>
<td>5242.23</td>
</tr>
<tr>
<td>Emacs 23.2</td>
<td>252</td>
<td>3800</td>
<td>1630</td>
<td>42.9</td>
<td>1802.24</td>
</tr>
<tr>
<td>Glib 2.24.0</td>
<td>236</td>
<td>6293</td>
<td>3020</td>
<td>48.0</td>
<td>3240.81</td>
</tr>
<tr>
<td>Cyrus imapd 2.3.13</td>
<td>225</td>
<td>1654</td>
<td>1150</td>
<td>68.2</td>
<td>1131.72</td>
</tr>
</tbody>
</table>
separation logic  abstract interpretation  bi-abduction

\[
\begin{array}{c}
\{I\} \ F \ \{E\} \\
N \vdash R \\
\{I \ast N\} \ F \ \{E \ast R\}
\end{array}
\]
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

People
Moving Fast

72 hours to launch Celebrate Pride

Omid Aziz  Derrick McMillen  Tony Liu
Perpetual Development
We don't want humans waiting on computers. We want computers waiting on humans.
Faster Deployment Model
CI system

Phabricator

Code reviewers

CI system

Developer

Product

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

Product

Developer

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!

You, Cristiano Calcagno, like this.
hi - Apologies for not addressing it earlier. I just commented, that it is a false positive because there is a check for it upstream.

Peter O'Hearn
Ah, that is pretty subtle. Thanks a lot for pointing that out!

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

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).
K-9 Android email client

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

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

```
p=OPENSSL_malloc(len);
BUF_strlcpy(p,t,len);
```
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 'cleanseCtr' 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;
}
size_t BUF_strlcpy(char *dst, const char *src, size_t size)
{
    size_t l = 0;
    for(; size > 1 && *src; size--)
    {
        *dst++ = *src++;
        l++;
    }
    if (size)
        *dst = '\0';
    return l + strlen(src);
In between true and false bugs

REMARKS:
— The definition of X509_gmtime_adj is in crypto/x509/x509_vfy.c. It calls X509_time_adj 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.
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);
}

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

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: Infer came up as a possible way to catch this. Have we worked with the 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.

\[ A \ast ?antiframe \vdash B \ast ?frame \]
Facebook Open-Sources Infer To Help Developers Identify Bugs Before They’re Shipped

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.

I'm posting from that very room right now.

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

My initial thoughts after about an hour of playing is the this is pretty bad ass
THIS JOURNEY
1% FINISHED
Don't (only) be whole program
Be Compositional