

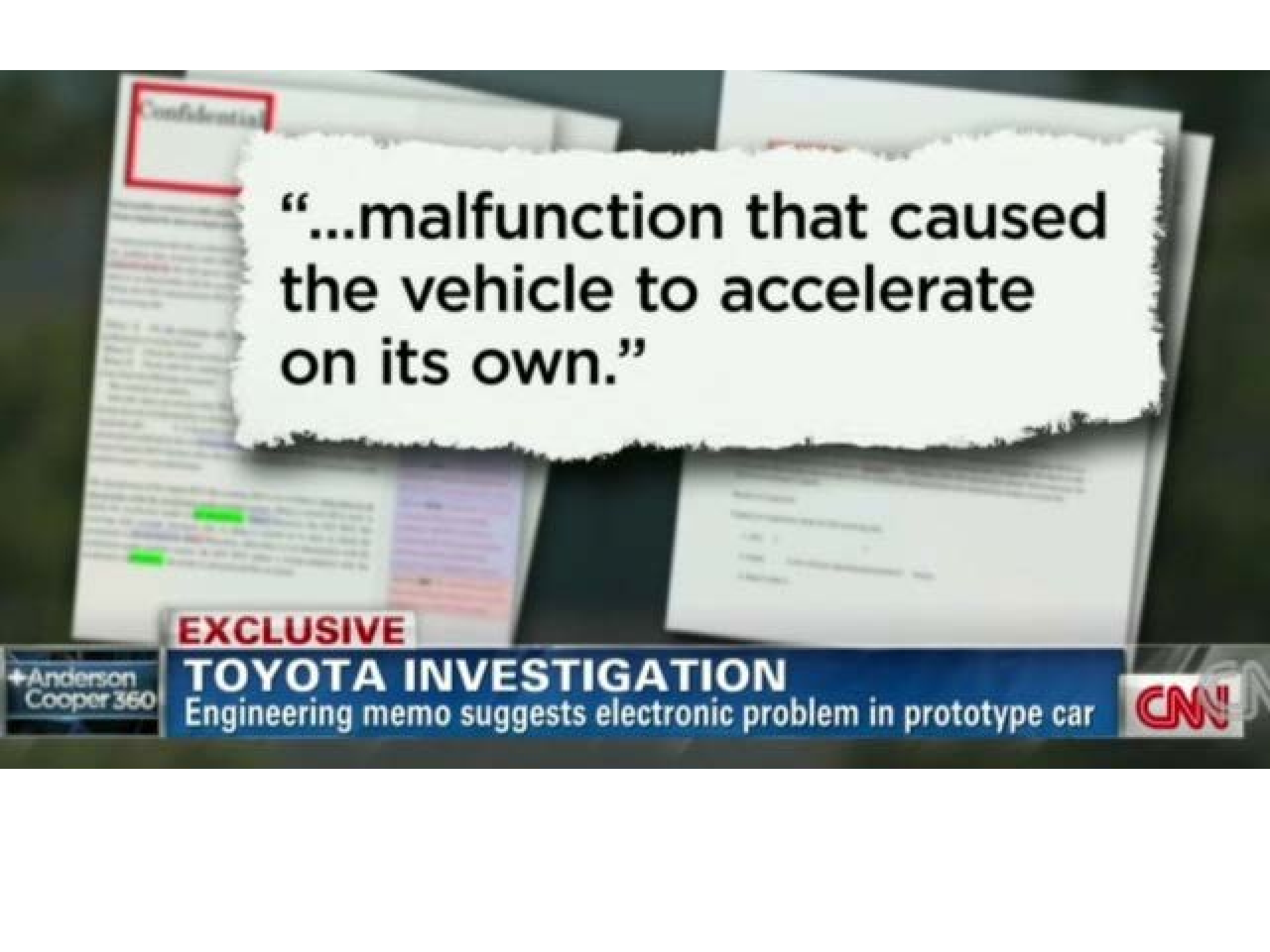
Software Technology

Magnus Myreen

(Using material from previous years, including material by David Sands)

MUCH TEST

VERY BUG



“...malfunction that caused the vehicle to accelerate on its own.”

EXCLUSIVE

TOYOTA INVESTIGATION

Engineering memo suggests electronic problem in prototype car

Anderson
Cooper 360

CNN

CHP Officer, Family Killed in Crash

A 911 call made minutes before the accident said the car's accelerator was stuck

By Rory Devine, Mari Payton and R. Stickney | Tuesday, Sep 1, 2009

[View Comments \(\)](#) | [Email](#) | [Print](#)



Source: <http://www.nbcsandiego.com/news/local/CHP-Oficer-Family-Killed-in-Crash-56629472.html>



An image taken from the air shows the vehicle resting in the brush just off the road.

2010

Over 6000 complaints of unintended acceleration

US Congress instigates NASA investigation

NASA Conclusions

- NASA didn't find a "smoking gun"
 - Tight timeline & limited information [Bookout 2013-10-14AM 39:18-40:8]
 - **Did *not* exonerate system**

Proof for the hypothesis that the ETCS-i caused the large throttle opening UAs as described in submitted VOQs could not be found with the hardware and software testing performed.

Because proof that the ETCS-i caused the reported UAs was not found does not mean it could not occur. However, the testing and analysis described in this report did not find that TMC ETCS-i electronics are a likely cause of large throttle openings as described in the VOQs.

[NASA UA Report, Executive Summary]

- But, U.S. Transportation Secretary Ray LaHood said,
 - “We enlisted the best and brightest engineers to study Toyota’s electronics systems, and the verdict is in. There is **no electronic-based cause** for unintended high-speed acceleration in Toyotas.”

<http://www.ntttsa.gov/PR/DOT-16-11>

News & Analysis

Toyota Case: Single Bit Flip That Killed

Junko Yoshida

10/25/2013 03:35 PM EDT

104 comments



14 saves

[LOGIN TO RATE](#)

During the trial, embedded systems experts who reviewed Toyota's electronic throttle source code testified that they found Toyota's source code defective, and that it contains bugs -- including bugs that can cause unintended acceleration.

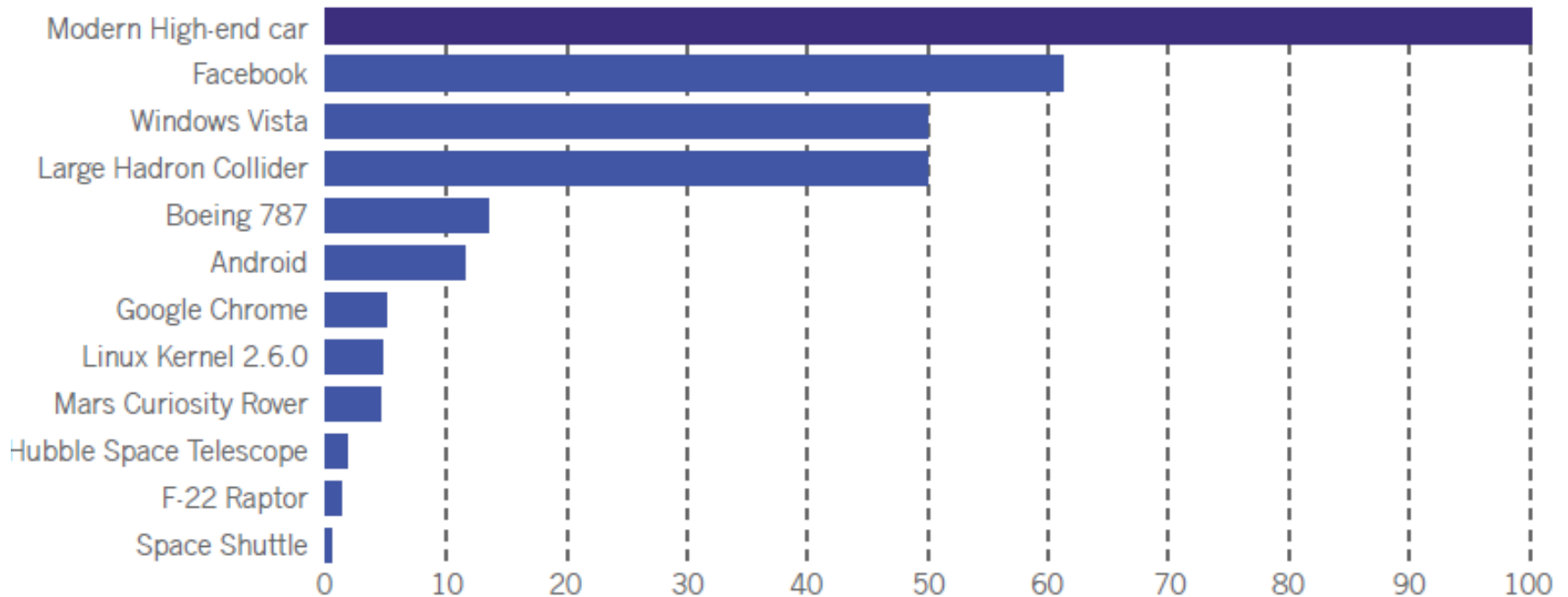
"We've demonstrated how as little as a single bit flip can cause the driver to lose control of the engine speed in real cars due to software malfunction that is not reliably detected by any fail-safe," Michael Barr, CTO and co-founder of Barr Group, told us in an exclusive interview. Barr served as an expert witness in this case.

Stack overflow and software bugs led to memory corruption, he said. And it turns out that the crux of the issue was these memory corruptions, which acted "like ricocheting bullets."

Bugs per line of code?

SOFTWARE SIZE (MILLION LINES OF CODE)

Source: NASA, IEEE, Wired, Boeing, Microsoft, Linux Foundation, Ohio



Concurrent Programming

Natural programming model in

- embedded systems
- operating systems
- GUIs

But it is easy to get wrong!

Sequential program

```
int counter = 0;
```

```
for(int i=0; i<1000000;i++) {  
    counter++;  
}
```

Concurrent Program

```
int counter = 0;
```

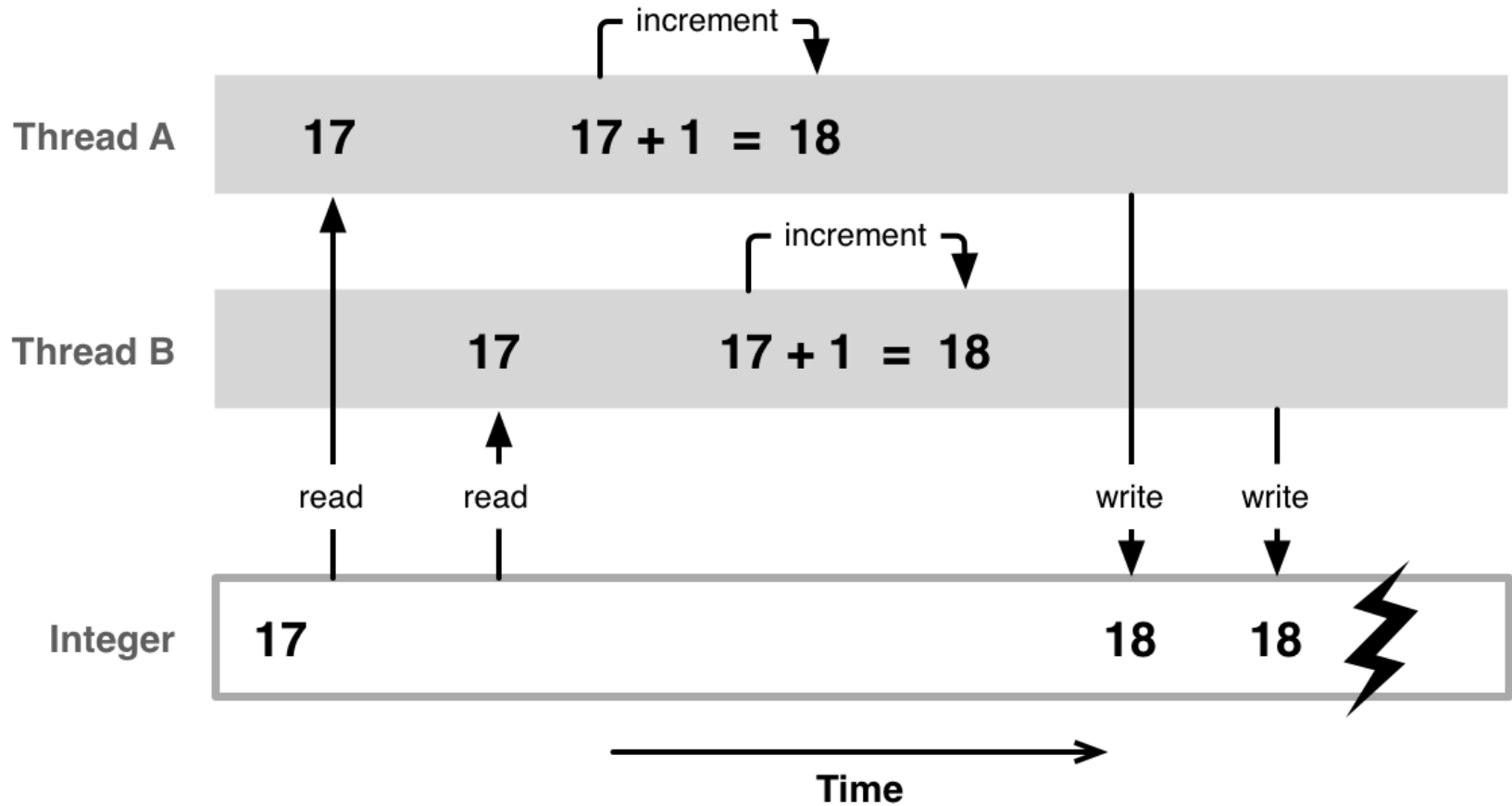
```
for(int i=0; i<1000000;i++) {  
    counter++;  
}
```

```
for(int i=0; i<1000000;i++) {  
    counter++;  
}
```

Demo

```
class Race implements Runnable {  
  
    int counter = 0;  
  
    public void run() {  
        for(int i=0; i<1000000;i++) { counter++; }  
    }  
  
    public static void main(String[] args) {  
        try {  
            Race r = new Race();  
            Thread A = new Thread(r);  
            Thread B = new Thread(r);  
            A.start(); B.start(); // Start both threads  
            A.join(); B.join(); // Wait for them to finish  
            System.out.println("Final counter: " + r.counter);  
        }  
        catch (Exception e) { }  
    }  
}
```

Data Race



Learn More!

Concurrent Programming

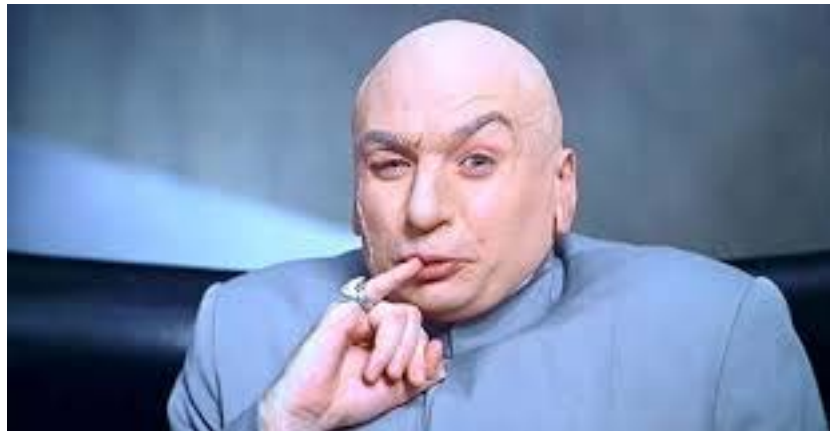
TDA384/DIT391 LP1, LP3

Testing, Debugging, and Verification

TDA567/DIT082, LP2

Bugs might make
things go wrong

will
Bugs ~~might~~ make
things go wrong



SECURITY WILL
RETURN IN
5 MINUTES

Please ring bell. If no response, page guard at
200-0100 and page 1119
T-1

FAIL
FAILBLOG.ORG

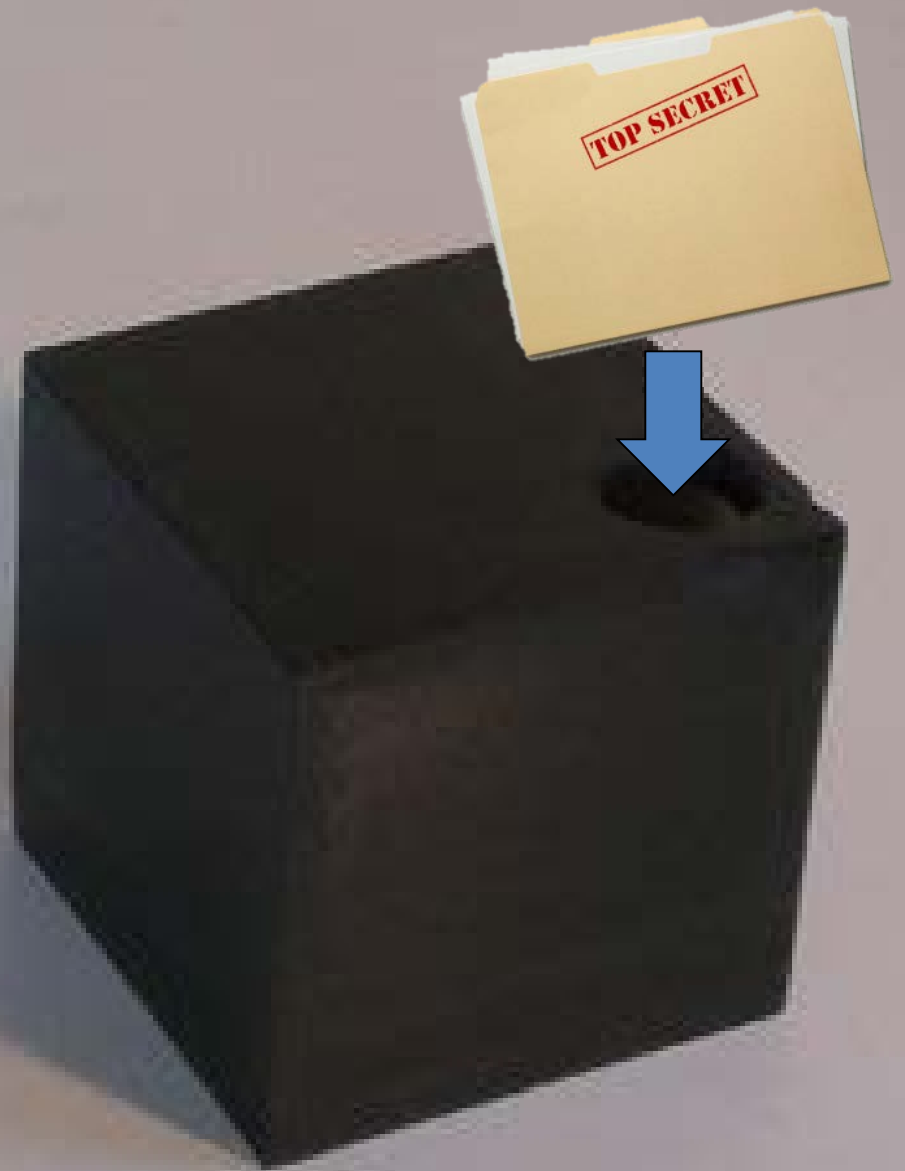
No bugs = Secure?

No bugs = Secure?

Does the software treat our sensitive data in an appropriate way?

What Information Flow Controls do we want?

- Confidentiality, Privacy
 - Information about sensitive data cannot be deduced by observing public channels
- Integrity
 - Untrusted data should not influence the values sent on trusted channels
- Erasure
 - information is no longer available after use







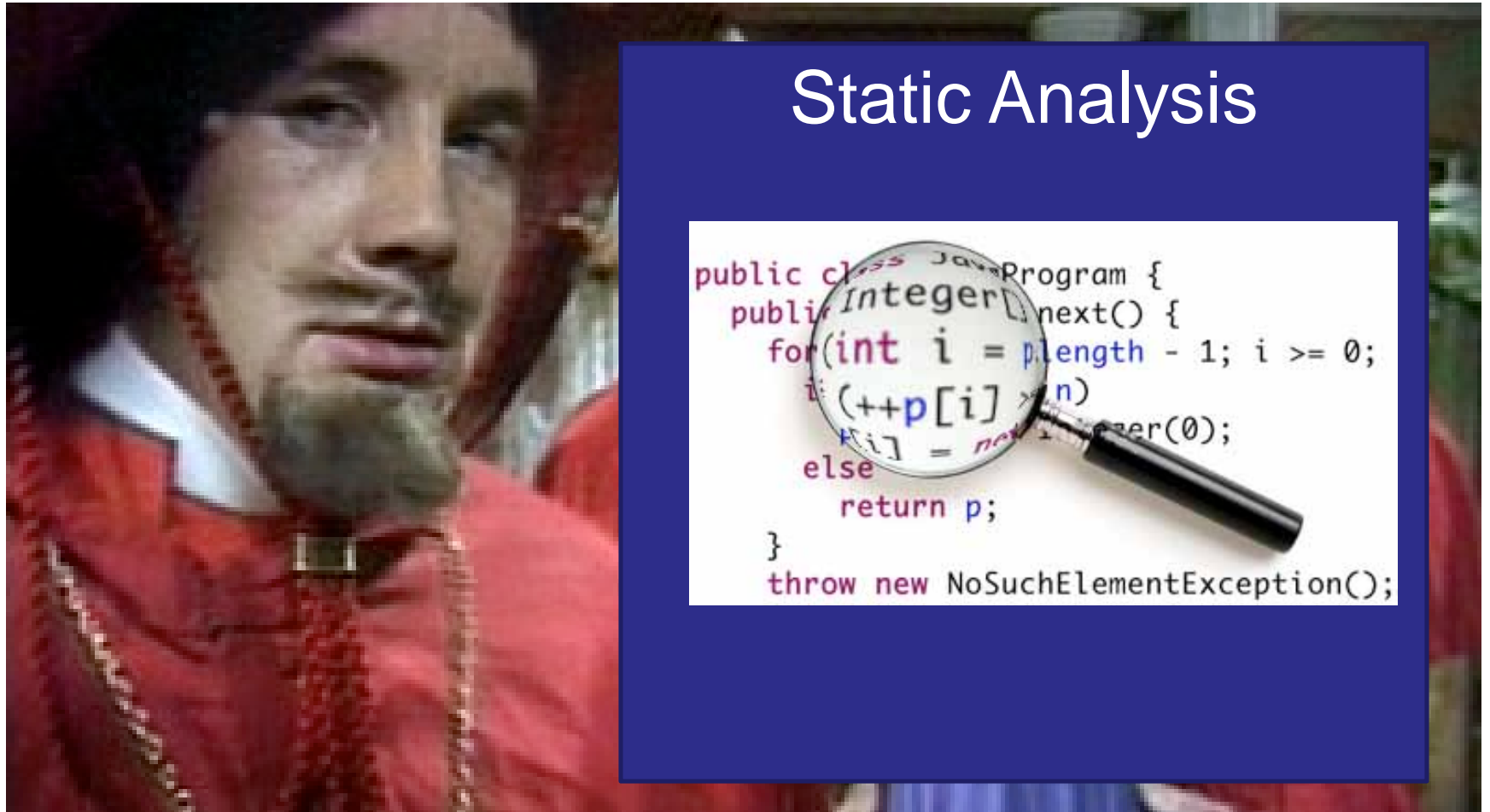
```
if (input != "attack at dawn")  
  { output("BANG!"); }
```

Our Chief Weapon



https://www.youtube.com/watch?v=Nf_Y4MbUCLY&t=15

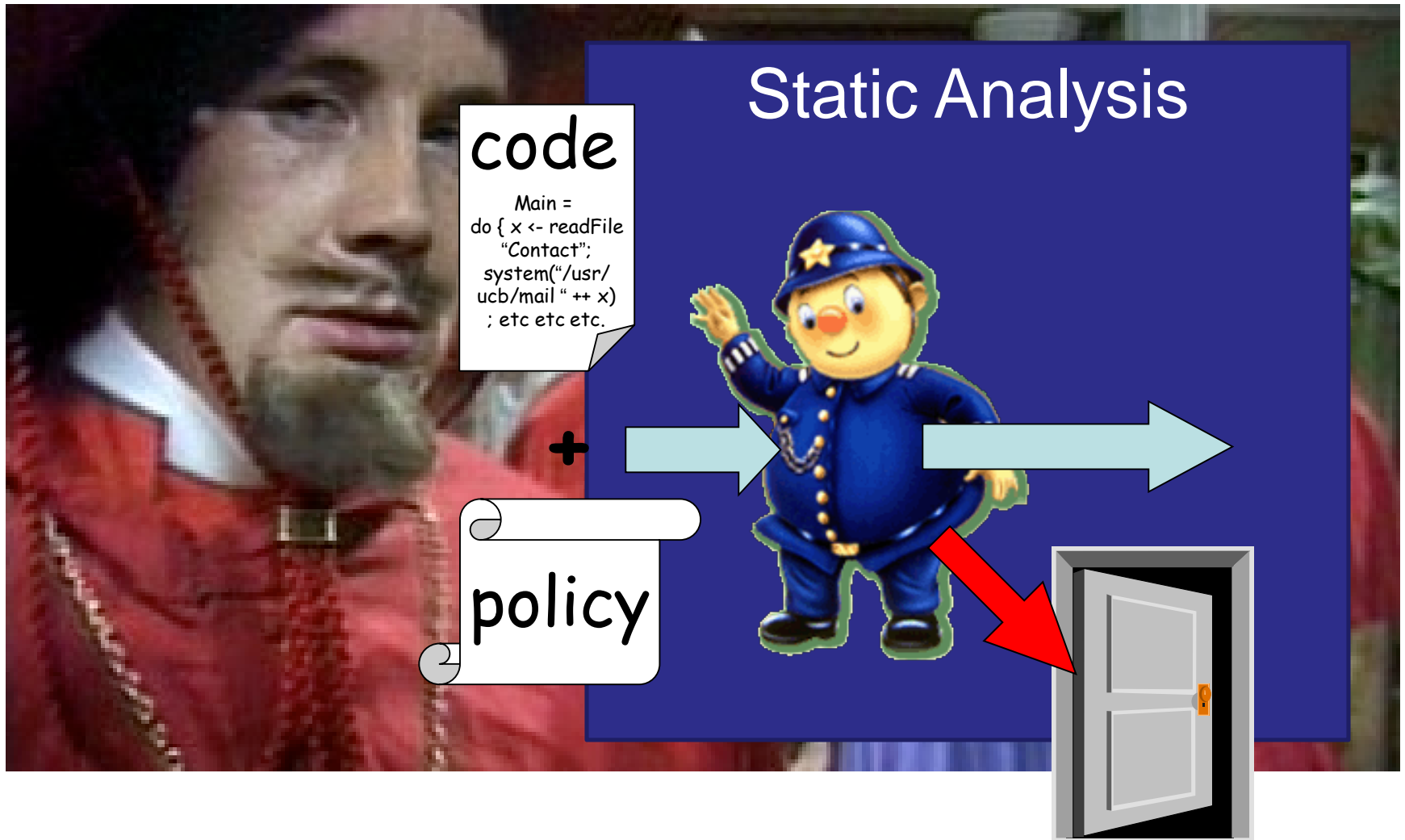
Our Chief Weapon



Static Analysis

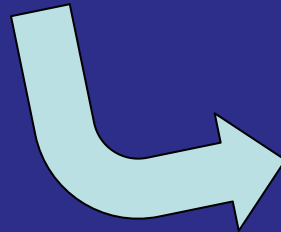
```
public class JavaProgram {  
    public Integer[] next() {  
        for(int i = p.length - 1; i >= 0;  
            i: (++p[i] > n)  
                i] = nextElement(0);  
            else  
                return p;  
        }  
        throw new NoSuchElementException();  
    }  
}
```

Our Chief Weapon

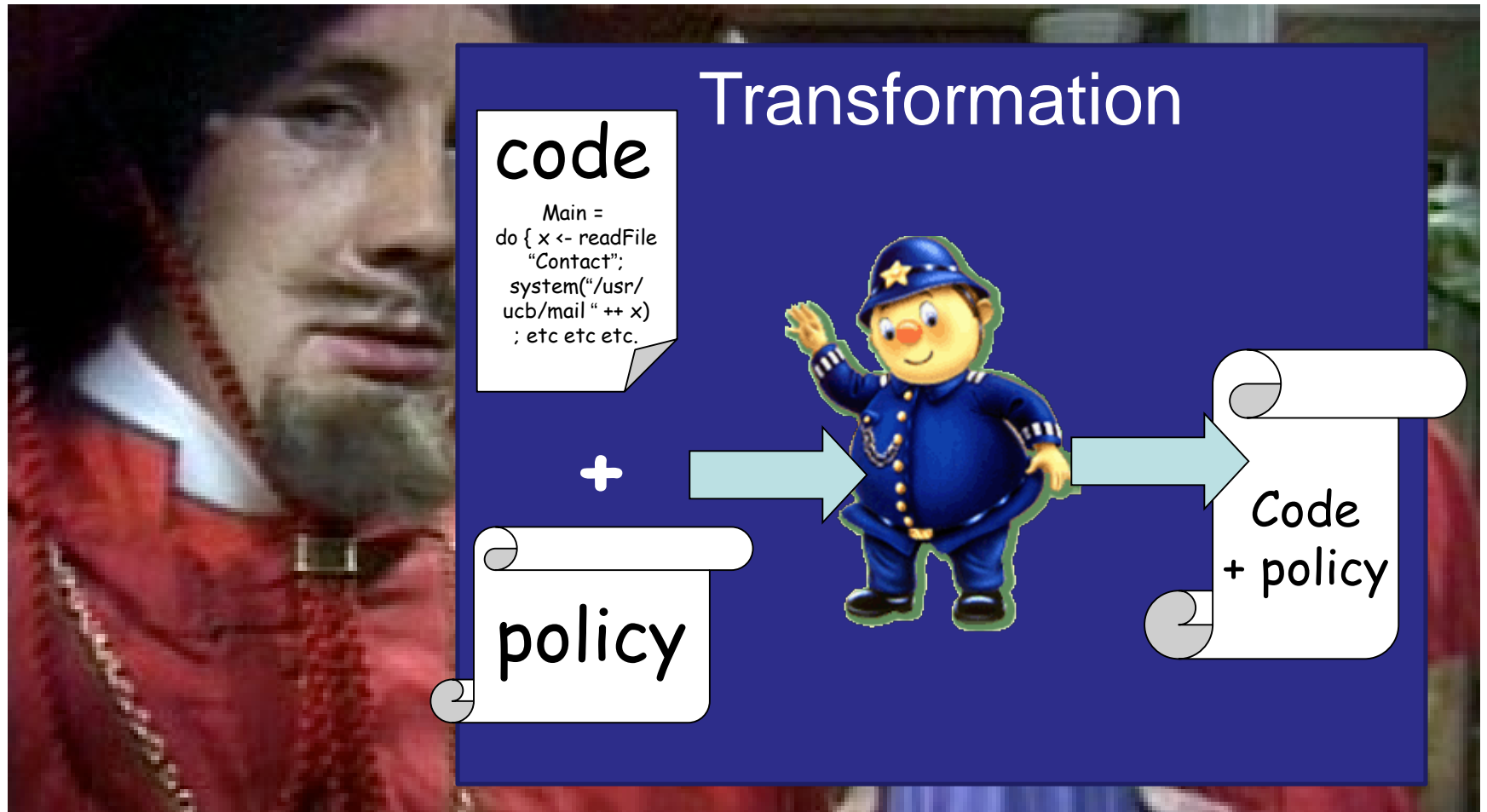


Our Chief Weapons

Transformation



Our Chief Weapons



Our Chief Weapons

Libraries

code

```
Main =  
do { x <- readFile  
  "Contact";  
  system("/usr/  
ucb/mail " ++ x)  
  ; etc etc etc.
```

+

policy
code

Our Chief Weapons

New Programming
Languages

code
policy

```
Main =  
do { x <- readFile  
    "Contact";  
    system("/usr/  
ucb/mail " ++ x)  
    ; etc etc etc.
```

Transformation
Static Analysis
Monitoring

[Home](#) >> [News](#) >> [Languages](#) >>

Main Menu

- [Home](#)
- [Book Reviews](#)
- [Book Watch](#)
- [News](#)
- [Projects](#)
- [The Core](#)
- [Babbage's Bag](#)
- [History](#)
- [Swift's Spreadsheets](#)
- [The Stone Tapes](#)
- [Professional Programmer](#)
- [eBooks](#)
- [Programmer Puzzles](#)
- [Bargain Computer Books](#)
- [CodeBin](#)
- [I Programmer Weekly](#)

Paragon - a programming language for security

Written by Kay Ewbank

Friday, 02 December 2011

A new programming language has been devised with the objective of plugging information leaks in software.

As many high profile stories of hackers obtaining information due to data leaks shows, it's not easy to make sure your application keeps its data safe. Researchers at the University of Gothenburg have developed a language that is designed to do the checks for you while you're writing your app



Paragon - a programming language for security

Written by Kay Ewbank

Friday, 02 December 2011

A new programming language has been devised with the objective of plugging information leaks in software.

As many high profile stories of hackers obtaining information due to data leaks shows, it's not easy to make sure your application keeps its data safe. Researchers at the University of Gothenburg have developed a language that is designed to do the checks for you while you're writing your app

1. PARAGON

The alternative, developed by Niklas Broberg at the University of Gothenburg is called Paragon, and the techniques used by the programming language are shown in his thesis *"Practical, Flexible Programming with Information Flow Control"*.

"The main strength of Paragon is its ability to automatically identify potential information leaks while the program is being developed,"

says Niklas Broberg.

New programming language to plug information leaks in software

NEWS: NOV

The current individuals have access to the code m Broberg of programmi information

Paragon identifies potential information leaks while the program is being written

As a solution to these problems, Niklas Broberg has developed the programming language Paragon. The methodology is presented in his thesis "Practical, Flexible Programming with Information Flow Control" which was written in August 2011.

"The main strength of Paragon is its ability to automatically identify potential information leaks while the program is being developed," says Niklas Broberg. "Paragon is an extension of the commonly-used programming language Java and has been designed to be easy to use. A programmer will easily be able to add my specifications to his or her Java program, thus benefiting from the strong security guarantees that the language provides."



What do we need to achieve this?

Deep understanding of programming language
design and implementation

Where to start?

Programming Language Technology

LP2 DAT151/DIT230

More to come (MSc)

- Compiler Construction TDA283/DIT300, LP4
- Language-based Security TDA602/DIT103, LP3

Courses

Concurrent programming

Testing, Debugging,
& Verification

Batchelor's level

Language-Based Security

Programming Language
Technology

Compiler Construction

Formal Methods for
Software Development

Master's level

... an error in java.util

```
Exception in thread "main"  
java.lang.ArrayIndexOutOfBoundsException: 40  
at java.util.TimSort.pushRun(TimSort.java:413)  
at java.util.TimSort.sort(TimSort.java:240)  
at java.util.Arrays.sort(Arrays.java:1438)  
at TestTimSort.main(TestTimSort.java:18)
```

... an error in java.util

```
Exception in thread "main"  
java.lang.ArrayIndexOutOfBoundsException: 40  
at java.util.TimSort.pushRun(TimSort.java:413)  
at java.util.TimSort.sort(TimSort.java:240)  
at java.util.Arrays.sort(Arrays.java:1438)  
at TestTimSort.main(TestTimSort.java:18)
```

Proving that Android's, Java's and Python's sorting algorithm is broken (and showing how to fix it)

🕒 February 24, 2015 📁 Envisage ✍️ Written by Stijn de Gouw. 🧑 \$s

<http://www.envisage-project.eu/proving-android-java-and-python-sorting-algorithm-is-broken-and-how-to-fix-it/>

The KeY project



- KeY lets you specify the desired behaviour of your program in the well-known specification language JML, and helps you prove that your programs conforms to its specification. That way, you did not only show that your program behaves as expected for some set of test values - ***you proved that it works correctly for all possible values!***
- Wolfgang Ahrendt (Chalmers) and others

A brief demo of KeY

<https://www.key-project.org/>



**KEEP
CALM
IT IS
DEMO
TIME**

Establishing Compiler Correctness

Maybe it is worth the cost?

Cost reduction?

Alternatives

- Proving the correctness of a compiler is prohibitively expensive (however, see the CompCert project)
- Testing is the only viable option

... but with testing you never know you caught all bugs!

All (unverified) compilers have bugs

“ Every compiler we tested was found to crash and also to silently generate wrong code when presented with valid input.”

PLDI'11

Finding and Understanding Bugs in C Compilers

Xuejun Yang Yang Chen Eric Eide John Regehr
School of Computing

“ [The verified part of] CompCert is the only compiler we have tested for which Csmith cannot find wrong-code errors. This is not for lack of trying: we have devoted about six CPU-years to the task.”

In this paper, we report on a large-scale study. Our first contribution is to advance the state of the art in compiler testing. Unlike previous tools, Csmith generates programs that cover a large subset of C while avoiding the common bugs that would destroy its ability

was heavily patched; the base version for Csmith

We created Csmith, a randomized test-case generator that supports testing Csmith gen-

Scaling up...

Project lead: Magnus Myreen
(now at Chalmers)

CakeML: Verified Implementation of ML

Ramana Kumar^{□ 1}

Magnus O. Myreen^{† 1}

Michael Norrish²

Scott Owens³

¹ Computer Laboratory, University of Cambridge, UK

² Canberra Research Lab, NICTA, Australia[‡]

³ School of Computing, University of Kent, UK

Abstract

We have developed and mechanically verified an ML system called CakeML, which supports a substantial subset of Standard ML. CakeML is implemented as an interactive read-eval-print loop (REPL) in x86-64 machine code. Our correctness theorem ensures that this REPL implementation prints only those results permitted by the semantics of CakeML. Our verification effort touches on a breadth of topics including lexing, parsing, type checking, incremental and dynamic compilation, garbage collection, arbitrary-precision arithmetic, and compiler bootstrapping.

Our contributions are twofold. The first is simply in building a system that is end-to-end verified, demonstrating that each of such a verification effort can in practice be composed of pieces that none of the pieces rely on any novel application of novel ap-

1. Introduction

The last decade has seen a strong interest in verified compilation; and there have been significant, high-profile results, many based on the CompCert compiler for C [1, 14, 16, 29]. This interest is easy to justify: in the context of program verification, an unverified compiler forms a large and complex part of the trusted computing base. However, to our knowledge, none of the existing work on verified compilers for general-purpose languages has addressed all aspects of a compiler along two dimensions: one, the compilation of a source string to a list of instructions; two, the execution of that

First bootstrapping of a formally verified compiler.

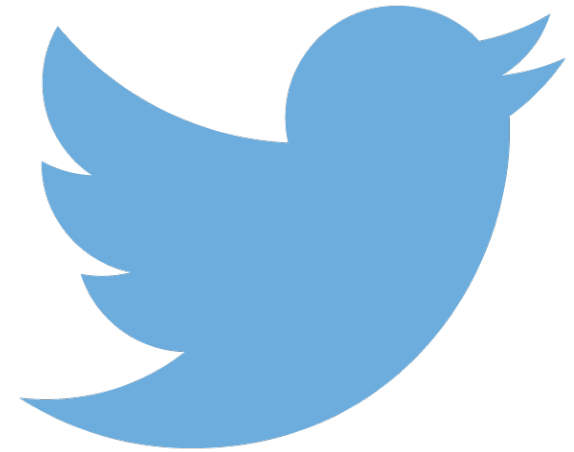
practically, we have implemented a standard ML and OCaml. By verifying the compilation of a source string to a list of instructions, we can ensure that the machine code along with the execution of that

... in a connected world:

... in a connected world:



Instagram





Dave Sands



12 hours ago with **Raúl Pardo** at **Chalmers Pub** · 🌐

Having some beers at the pub

Like · Comment · Share

 Devdatt and 20 people like this.



Gerardo Schneider Huh? Raúl is supposed to be working on tomorrow's presentation at FMPriv

Like · Reply ·  15 · 5 mins



Write a comment ...



HERE'S HOW TO USE FACEBOOK PRIVACY SETTINGS

What you need to know about privacy settings

How to make your posts private

SOCIAL MEDIA TIPS

Time to review your Twitter privacy settings to block annoying direct messages

You can now get private messages from anyone on Twitter, but you might want to turn this feature off

ITworld | April 27, 2015

PRIVACY NEWS

MORE GOOD READS

Twitter announces new privacy changes and...

ITWORLD

compare on privacy -- in one handy chart

RED



PRIVACY POLICIES



My supervisor cannot see my posts from 20:00 to 8:00



Where to start?

TDA294 / DIT271

Formal Methods for Software Development

(DAT060 / DIT201 Logic in computer science)

(DAT140 / DIT232 Types for Programs and Proofs)

All problems are not solved:



Information leakage due to speculation in hardware implementation.

2018: <https://meltdownattack.com/>