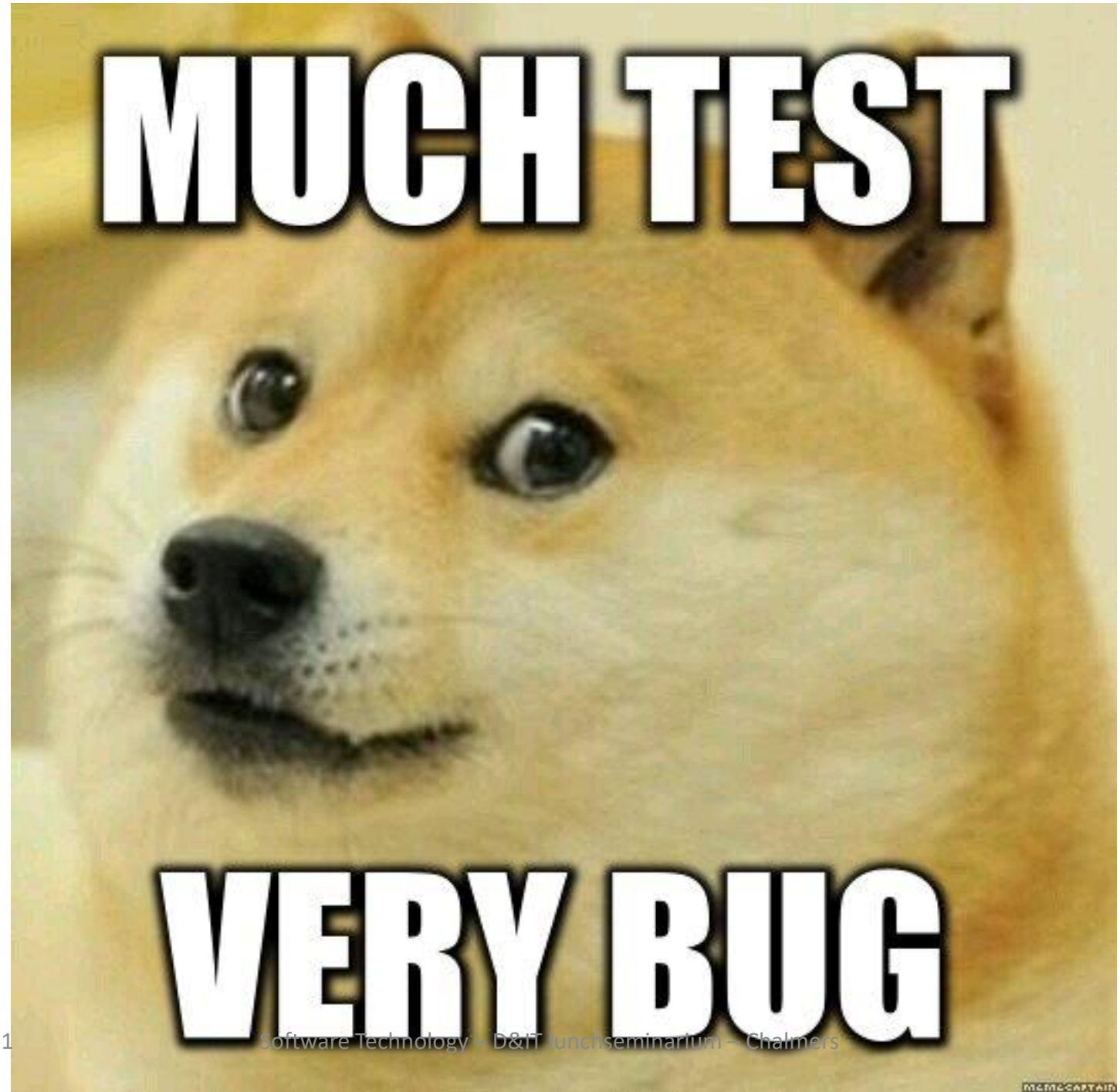
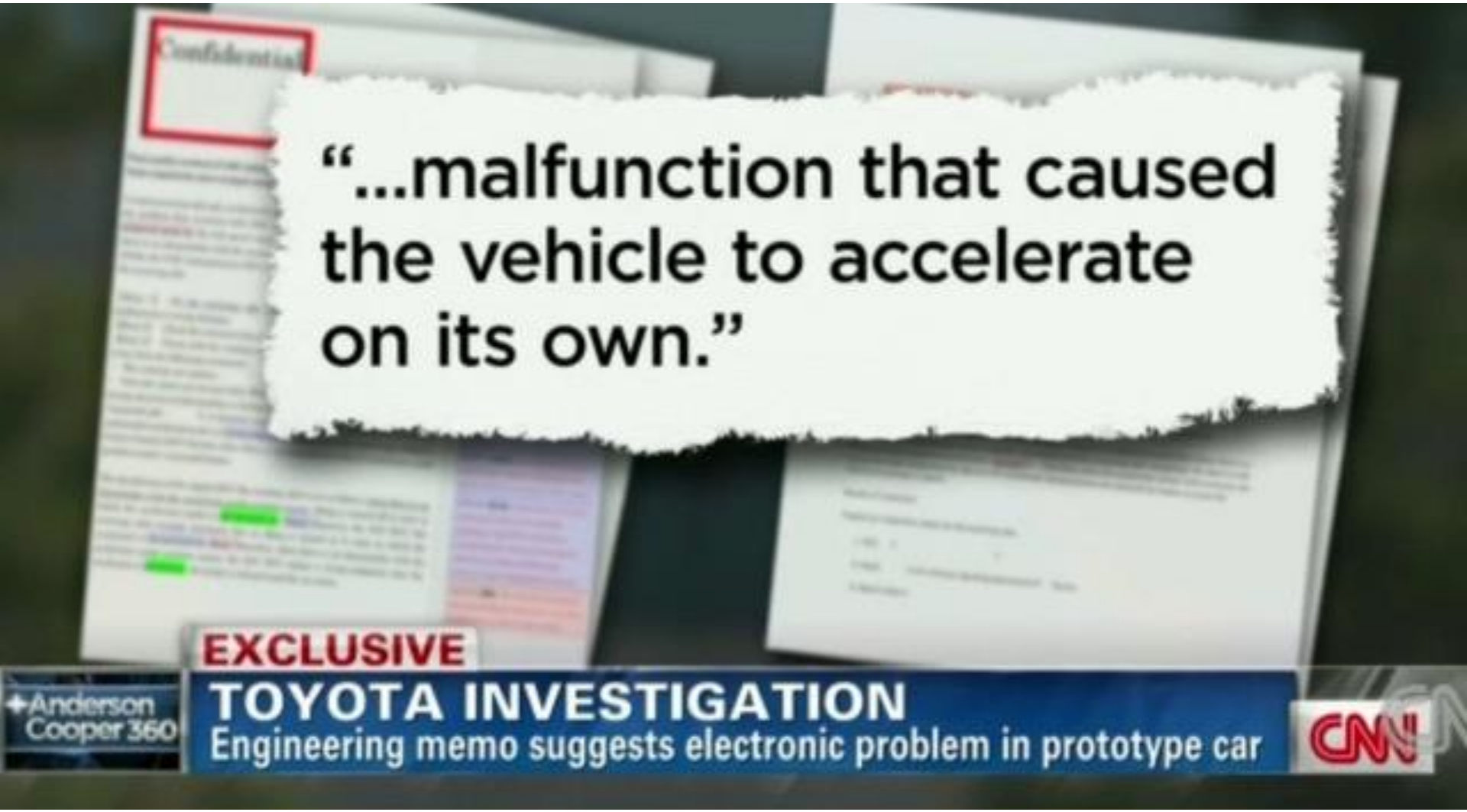
# Software Technology

Sandro Stucki

D&IT lunch seminar – 2019-11-21

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


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

7 SAN DIEGO

Source: http://www.nbcsandiego.com/news/local/CHP-Officer-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.nhtsa.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

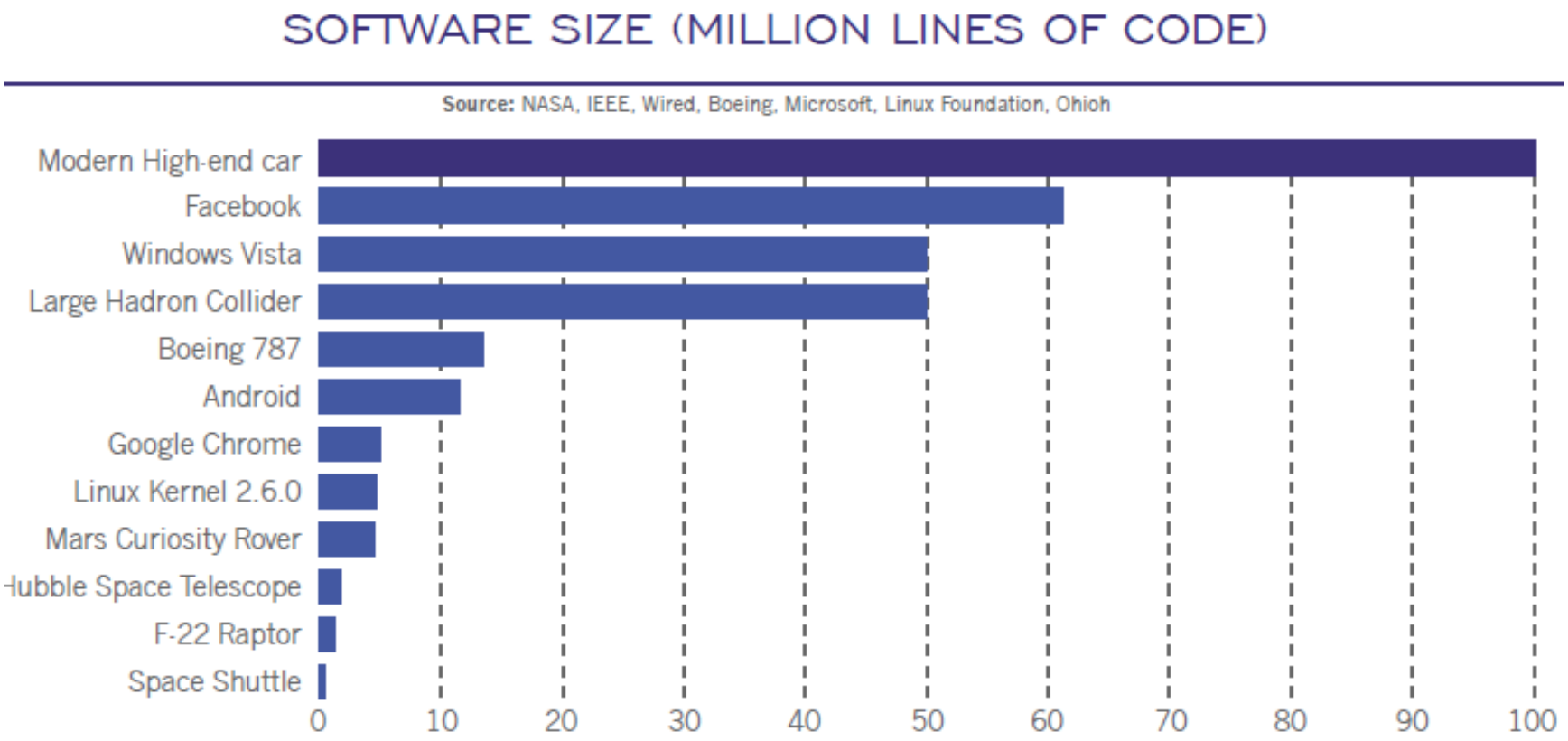


During the trial, embedded systems experts who reviewed Toyota's electronic throttle source code testified that they found Toyota's source code defective, and that it contains bugs -- including bugs that can cause unintended acceleration.

"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?



# 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

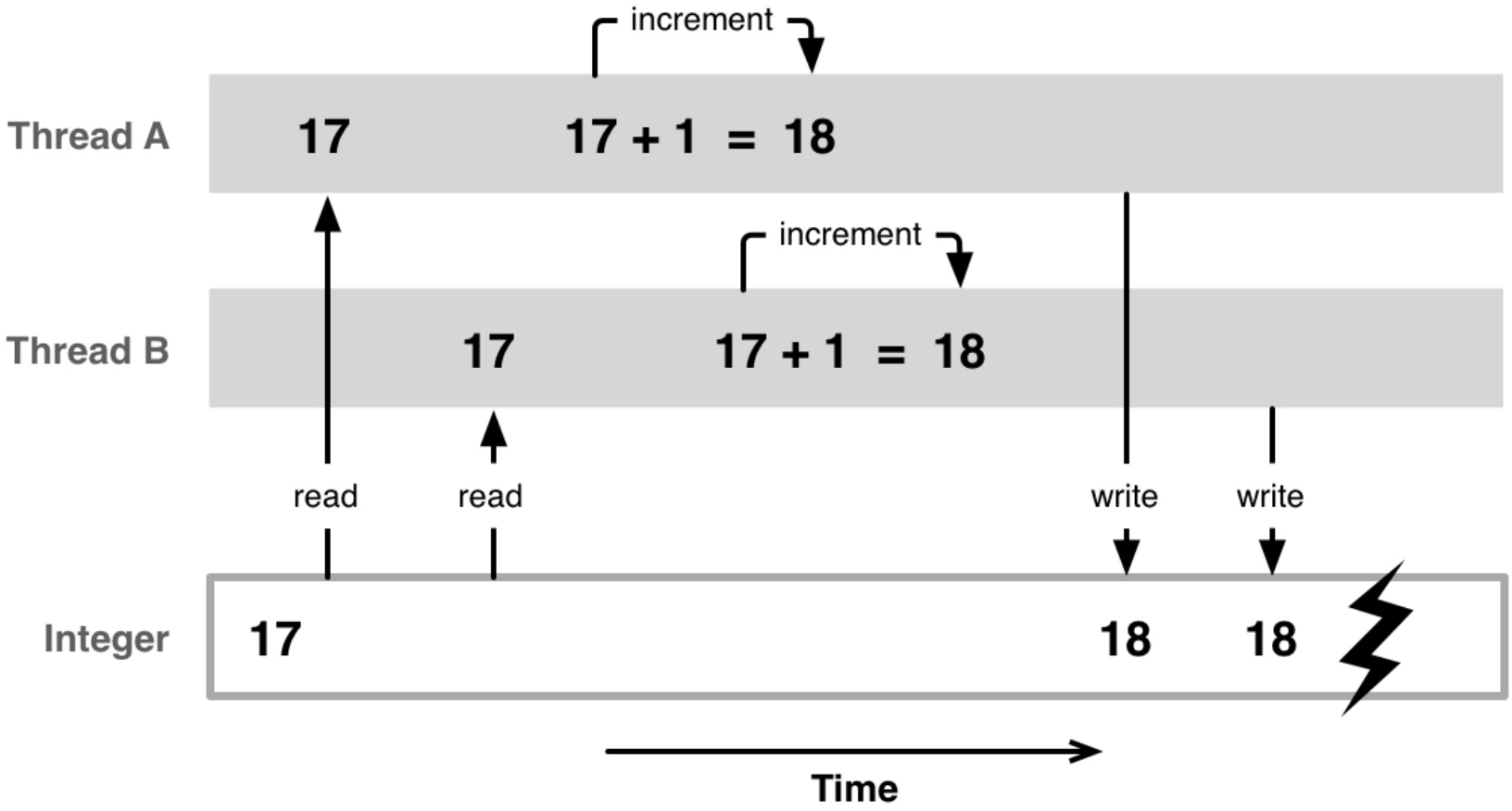
```
class Race implements Runnable {

    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


increment
Thread A
17
17 + 1 = 18
increment
Thread B
17
17 + 1 = 18
read
read
write
write
Integer
17
18
18
Time

# Learn More!

Concurrent Programming

TDA384/DIT391 LP1, LP3

Testing, Debugging, and Verification

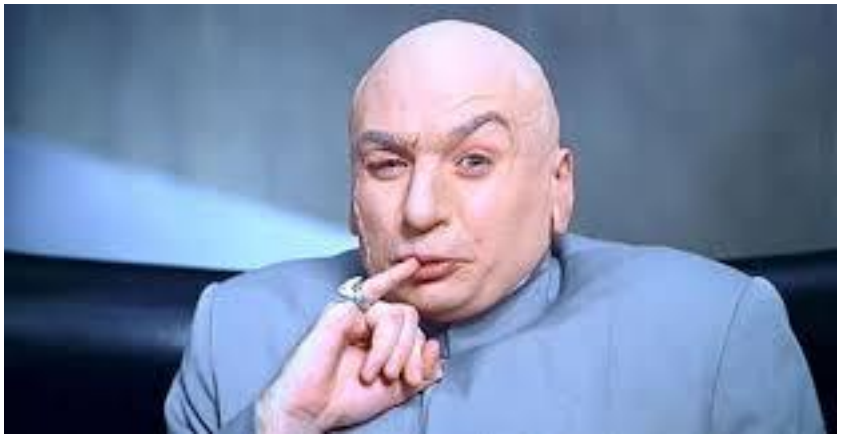
TDA567/DIT082, LP2

# Bugs might make things go wrong

will

Bugs might make

things go wrong




SECURITY WILL
RETURN IN
5 MINUTES
FAIL
FAILBLOG.ORG

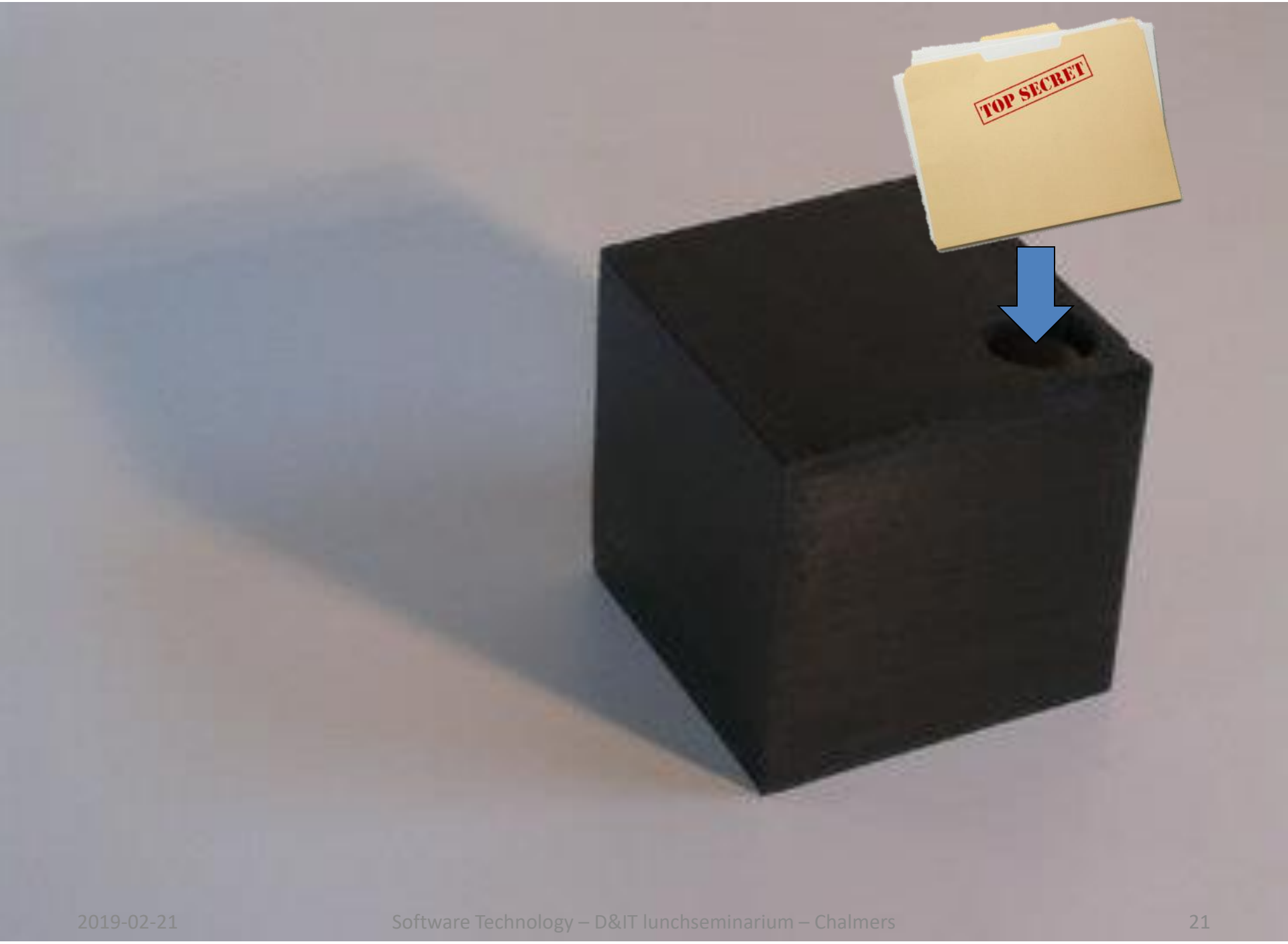
# No bugs = Secure?

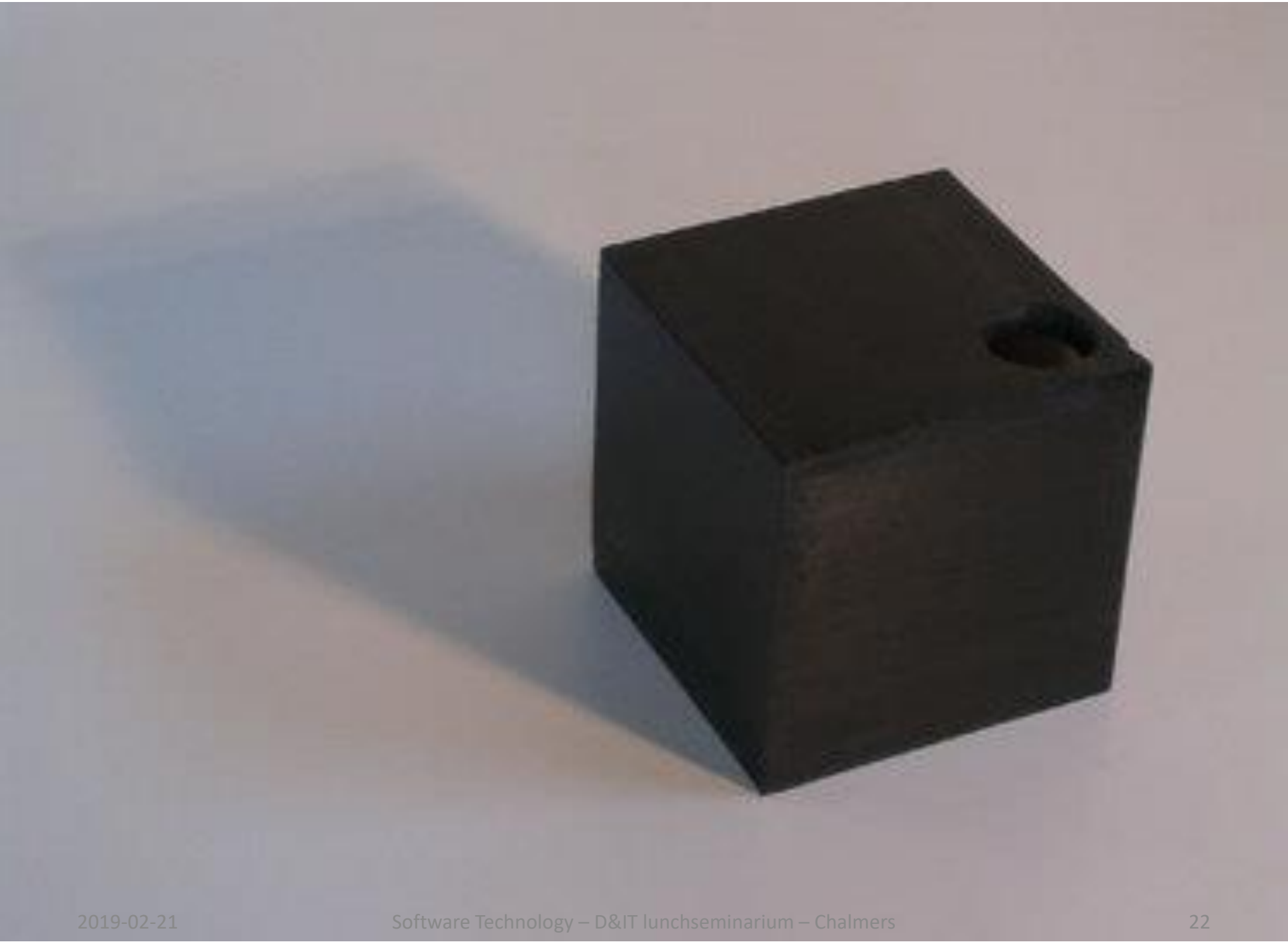
# No bugs = Secure?

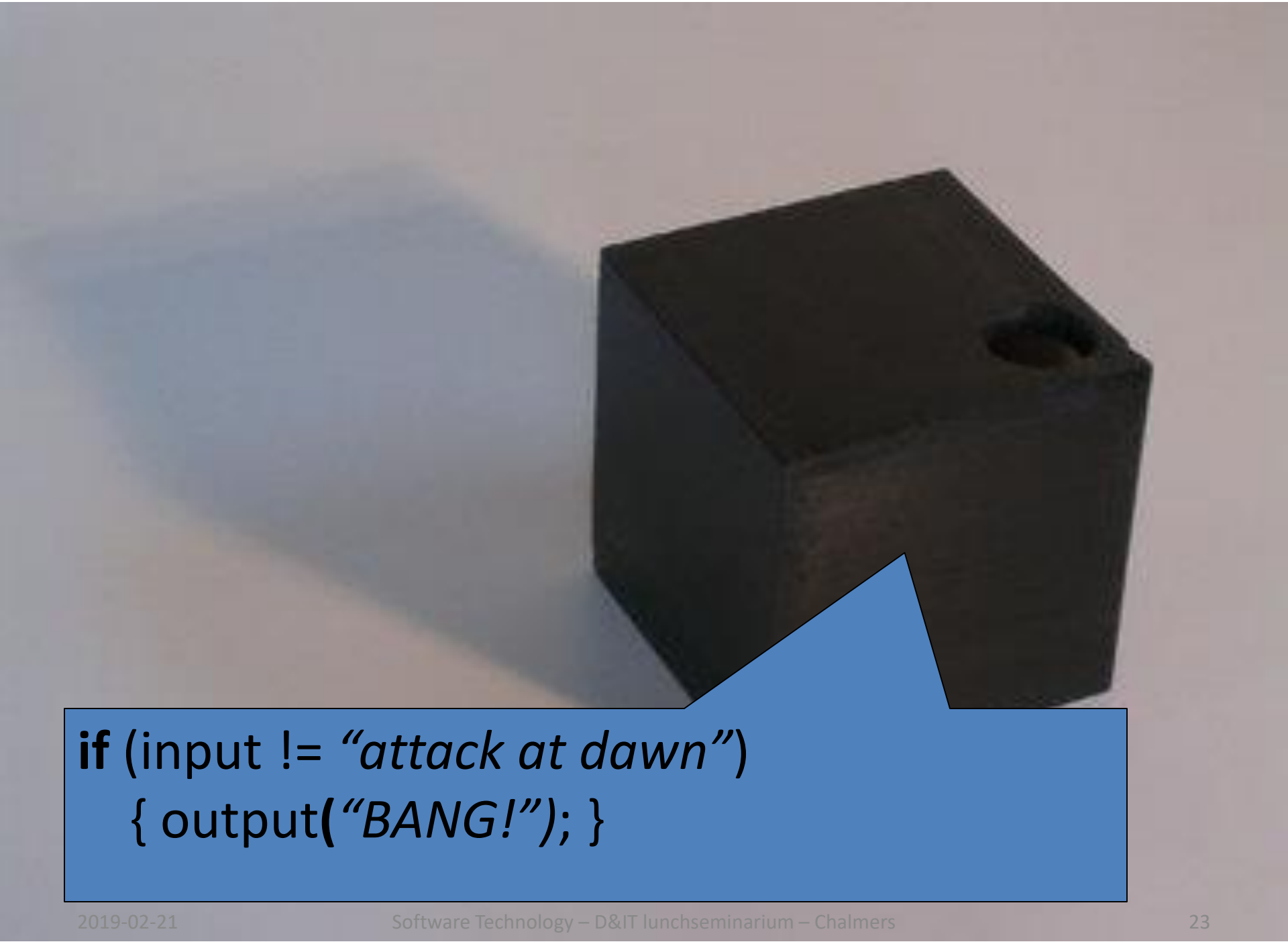
Does the software treat our sensitive data in an appropriate way?

# What Information Flow Control 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


TOP SECRET

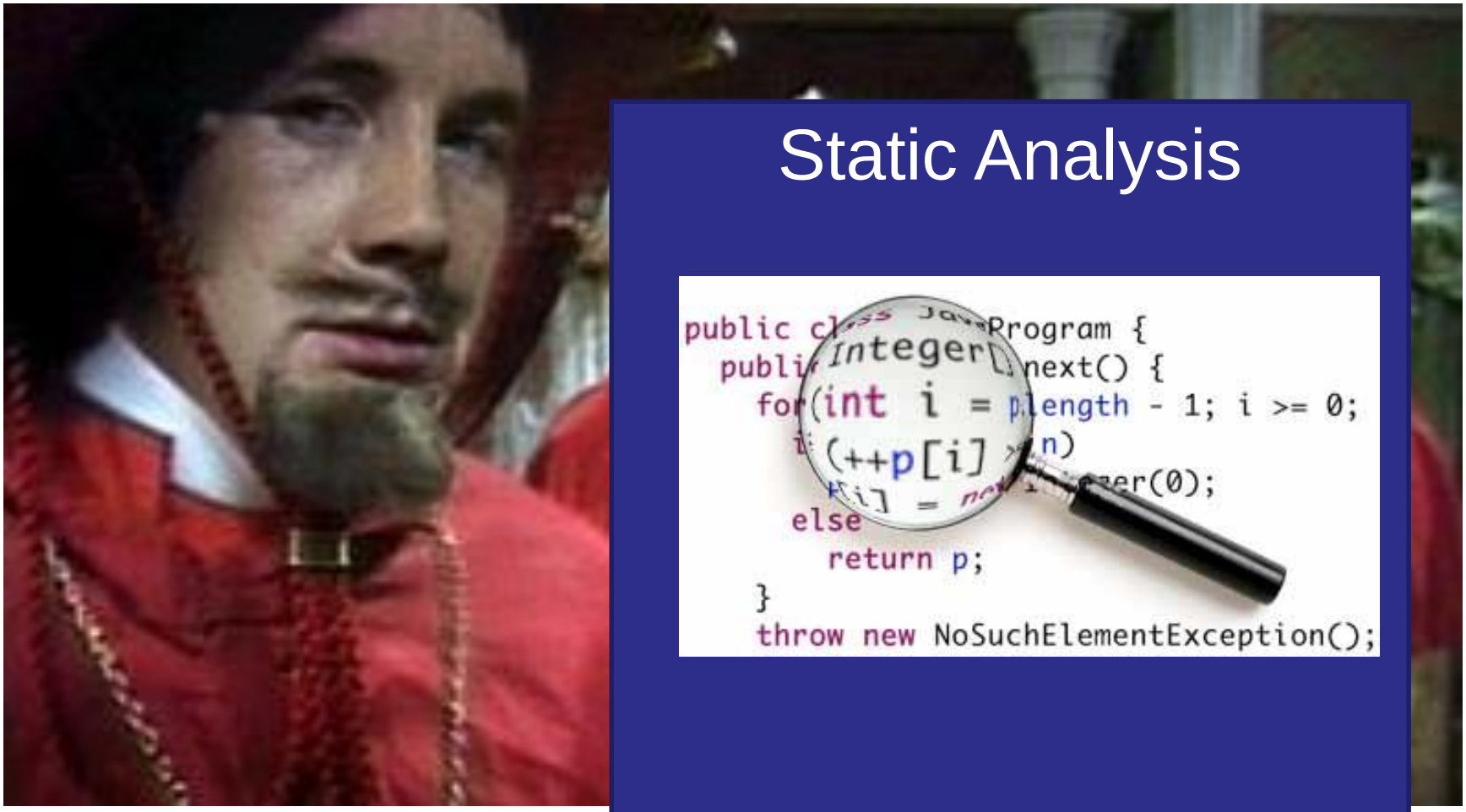



if (input != “attack at dawn”)
{ output(“BANG!”); }

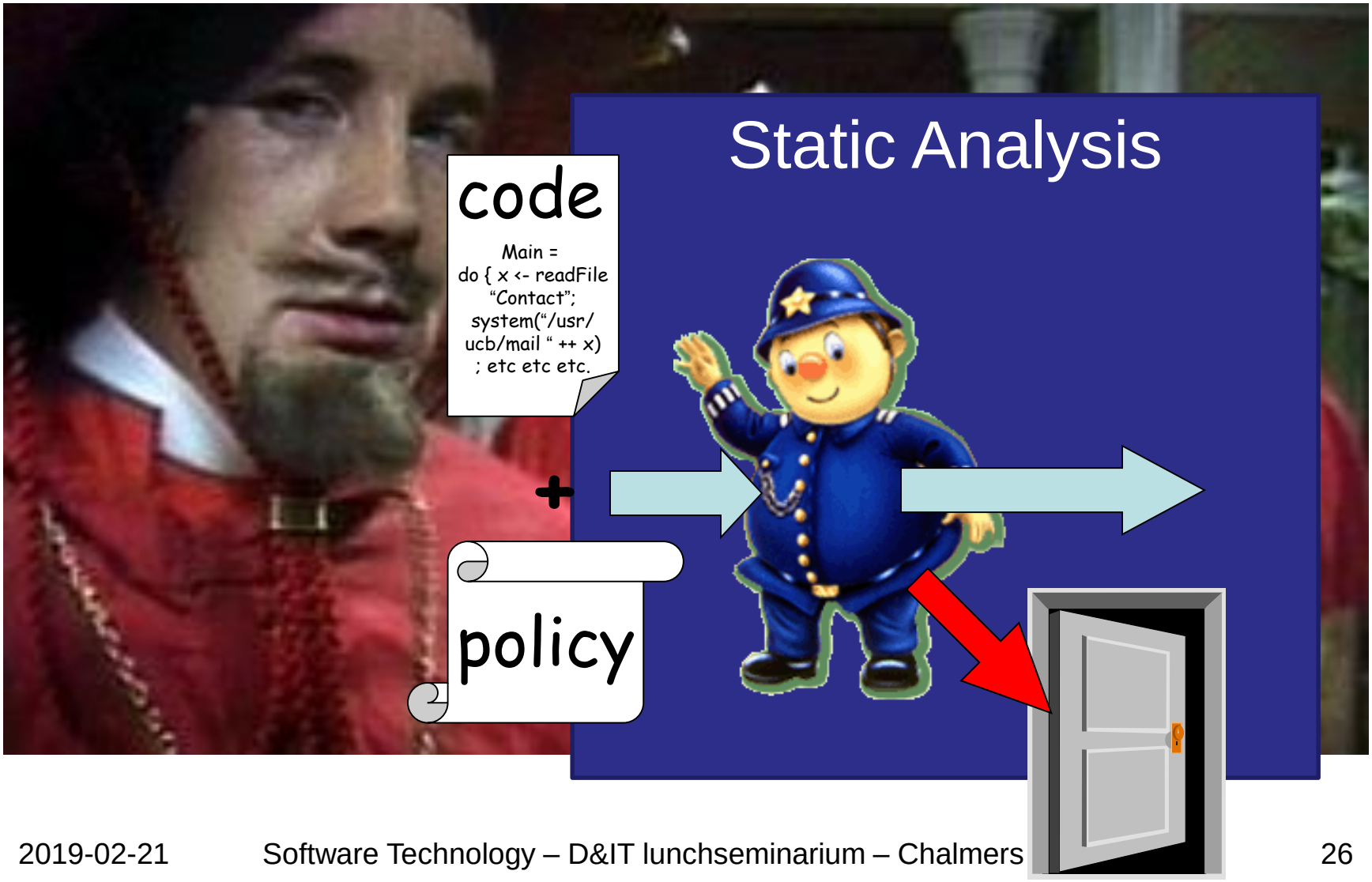
# Our Chief Weapon

https://youtu.be/1N6OOWtCYQA

# Our Chief Weapon

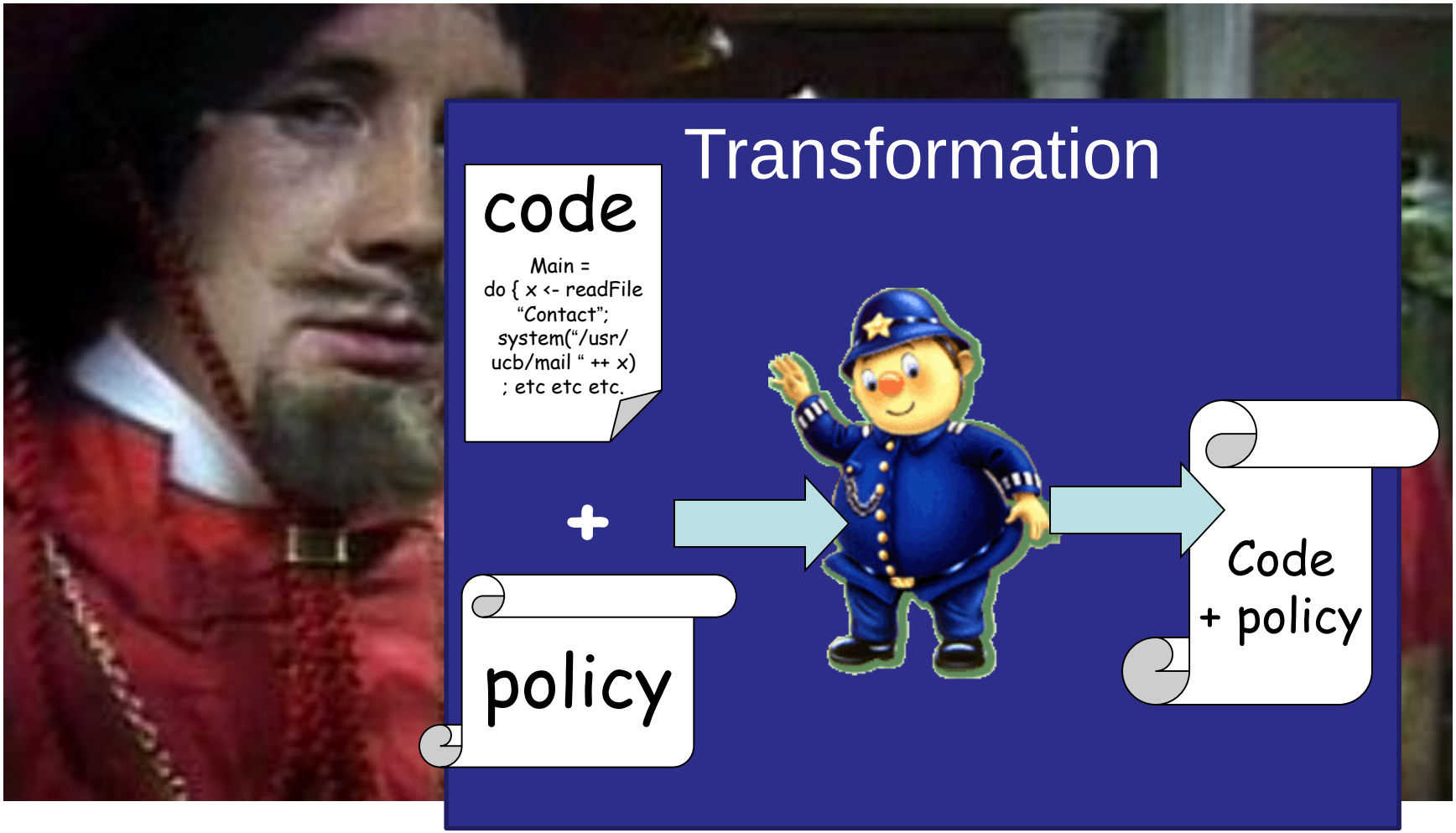


# Our Chief Weapon

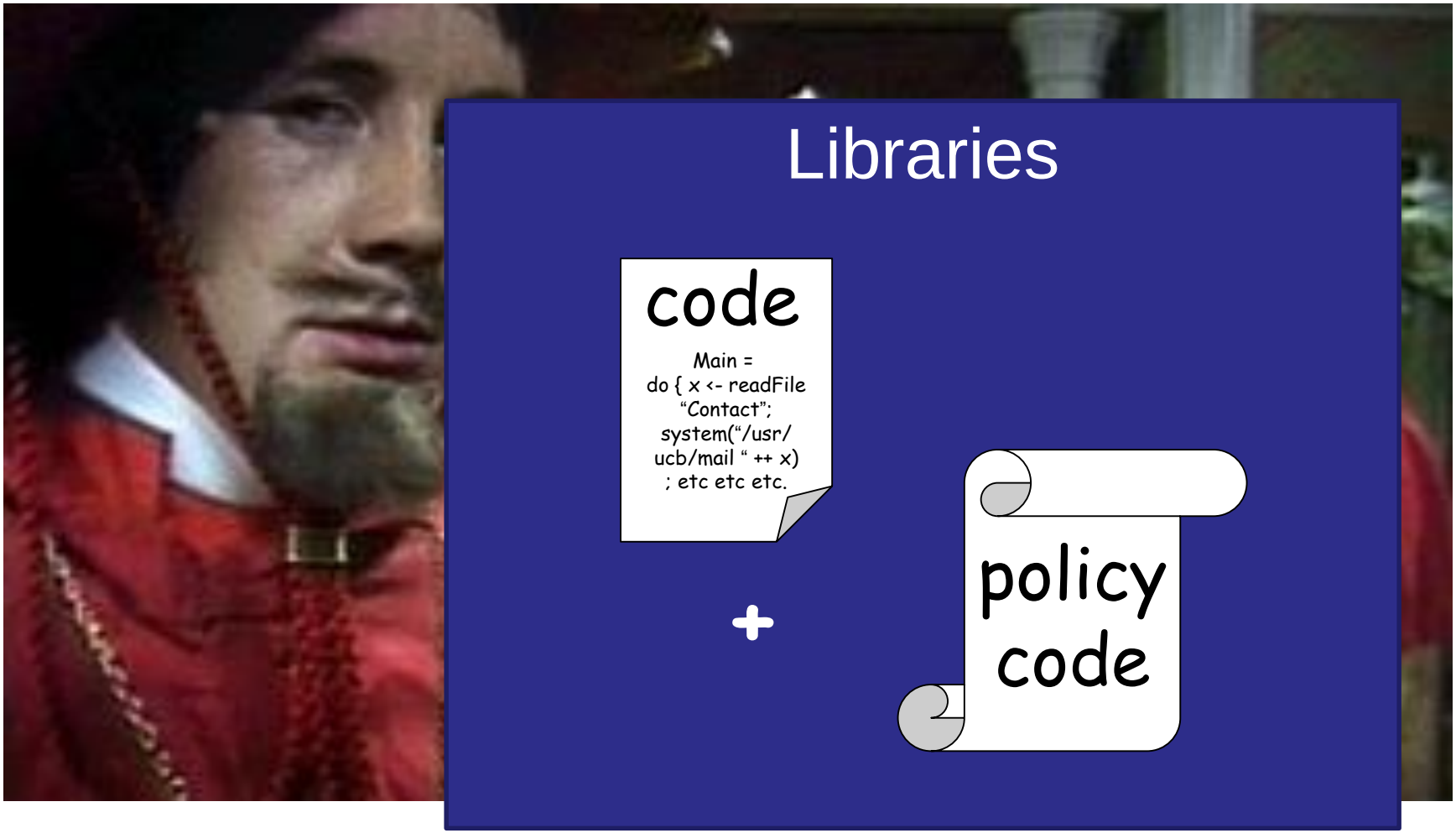


# Our Chief Weapons

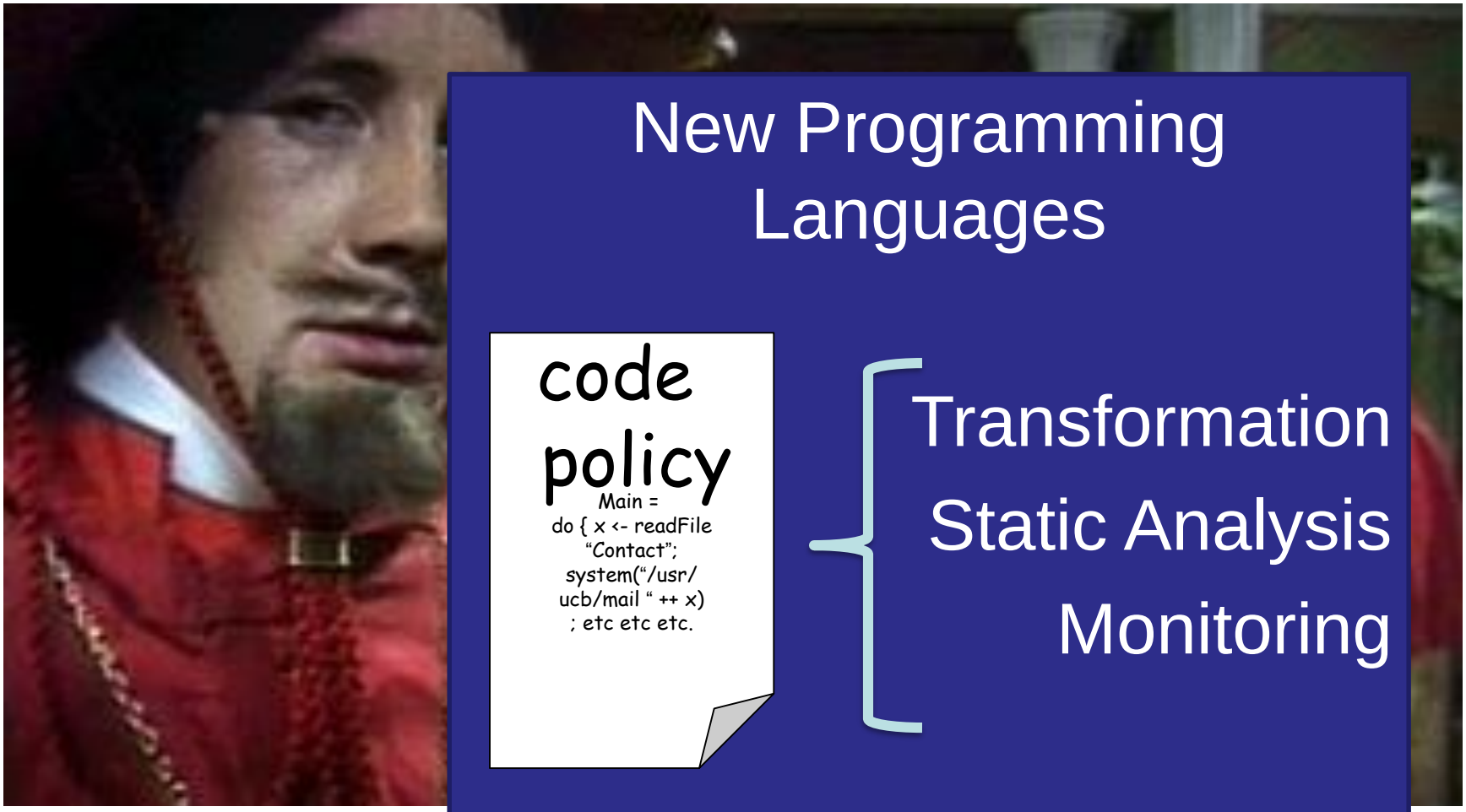
# Our Chief Weapons



# Our Chief Weapons



# 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

I PROGRAMMER

Login | Register | Edit Profile | Chang

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

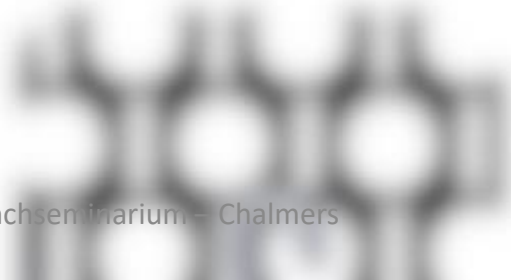
A new programming language has been devised with the objective of plugging information leaks in software.

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

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 acces**
**the code m**
**Broberg of**
**programmi**
**informatio**

**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/DIT101, LP4
- Formal Methods for Software Development TDA294/DIT271, LP1

# Courses

Concurrent programming

Testing, Debugging, & Verification

**Bachelor'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/

# More to come (MSc)

- Compiler Construction TDA283/DIT300, LP4
- Language-based Security TDA602/DIT103, LP3
- Formal Methods for Software Development TDA294/DIT271, LP1

# Courses

Concurrent programming

Testing, Debugging, & Verification

**Bachelor's level**

Language-Based Security

Programming Language Technology

Compiler Construction

Formal Methods for Software Development

**Master's level**

## Trusting the compiler

### Bugs

When finding a bug, we go to great lengths to find it in our own code.

- Most programmers trust the compiler to generate correct code
- The most important task of the compiler is to generate correct code

Maybe it is worth the cost?

Cost reduction?

## Establishing Compiler Correctness

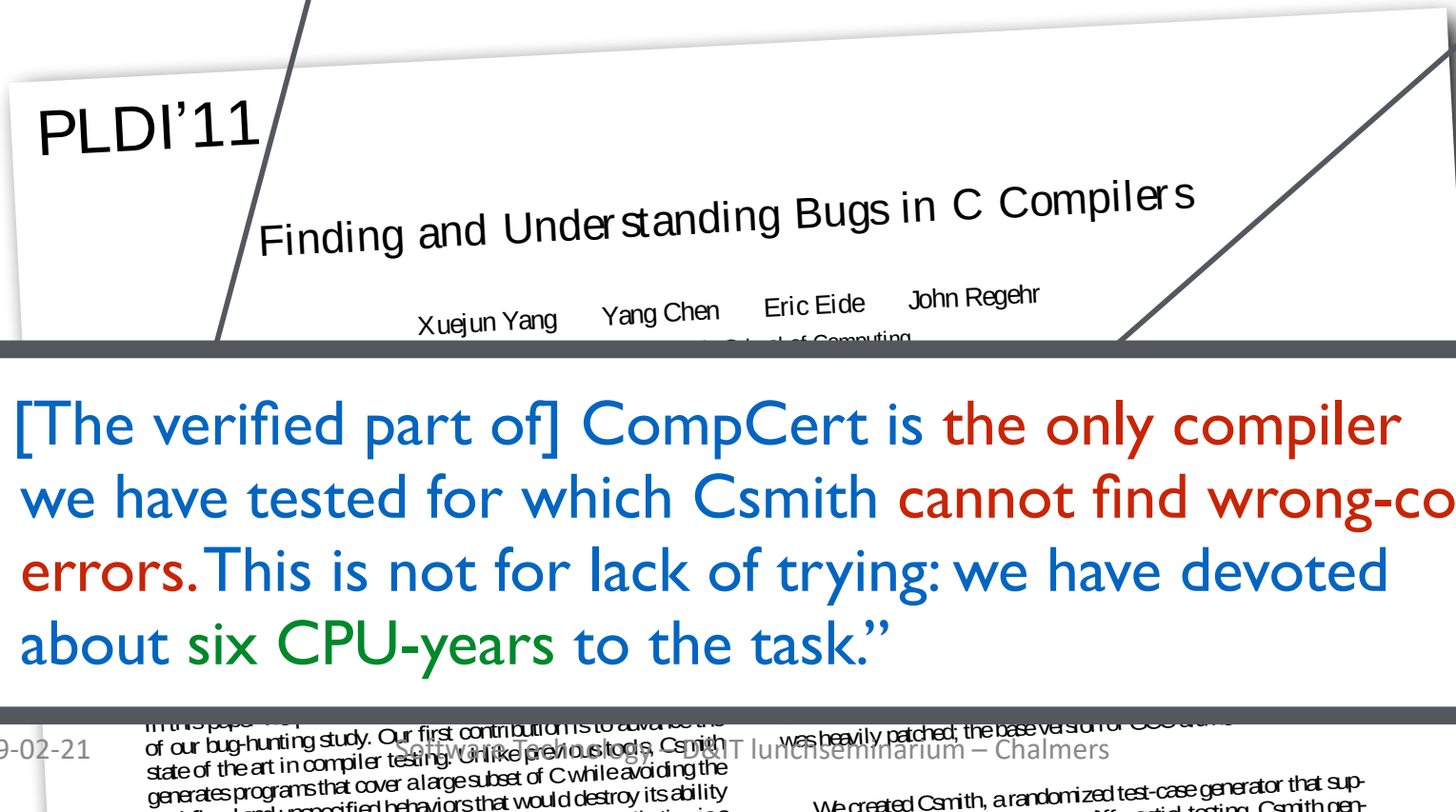
### Alternatives

- 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

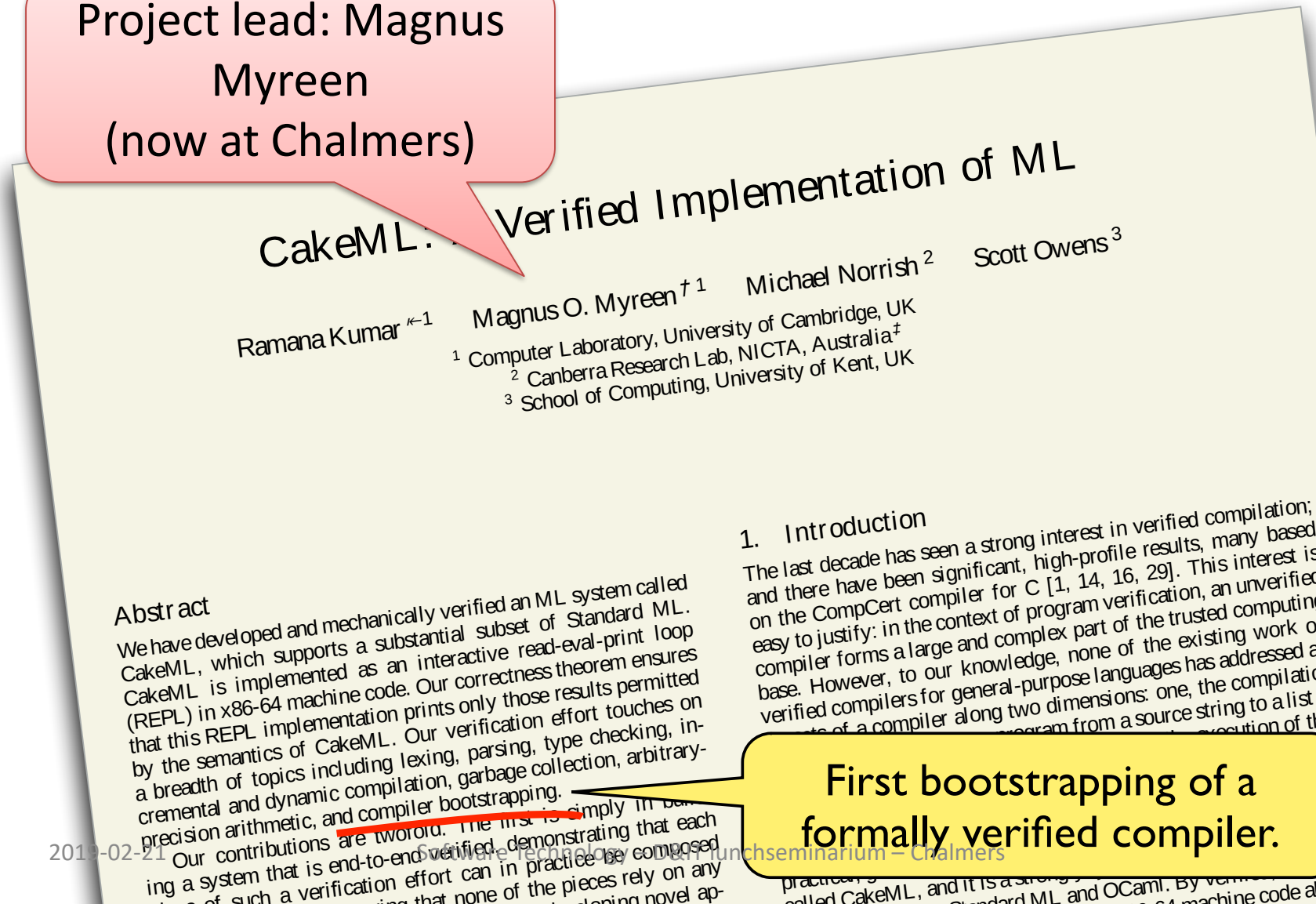
Finding and Understanding Bugs in C Compilers

Xuejun Yang Yang Chen Eric Eide John Regehr

" [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."

# Scaling up…

Project lead: Magnus Myreen (now at Chalmers)



CakeML: A Verified Implementation of ML

Ramana Kumar [1] Magnus O. Myreen [2] Michael Norrish [3] Scott Owens [4]

[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 two-fold. 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 ing that none of the pieces rely on any

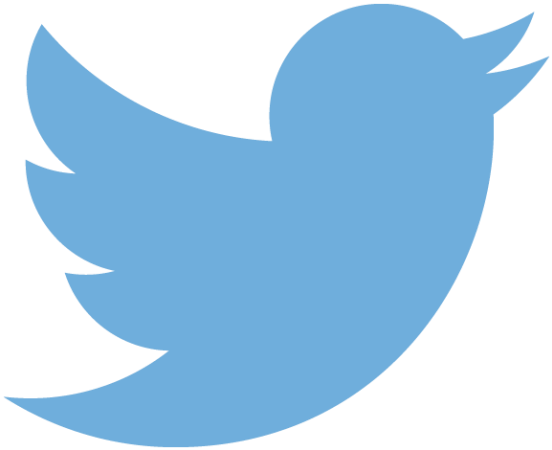
1. Introduction

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

First bootstrapping of a formally verified compiler.

# … 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

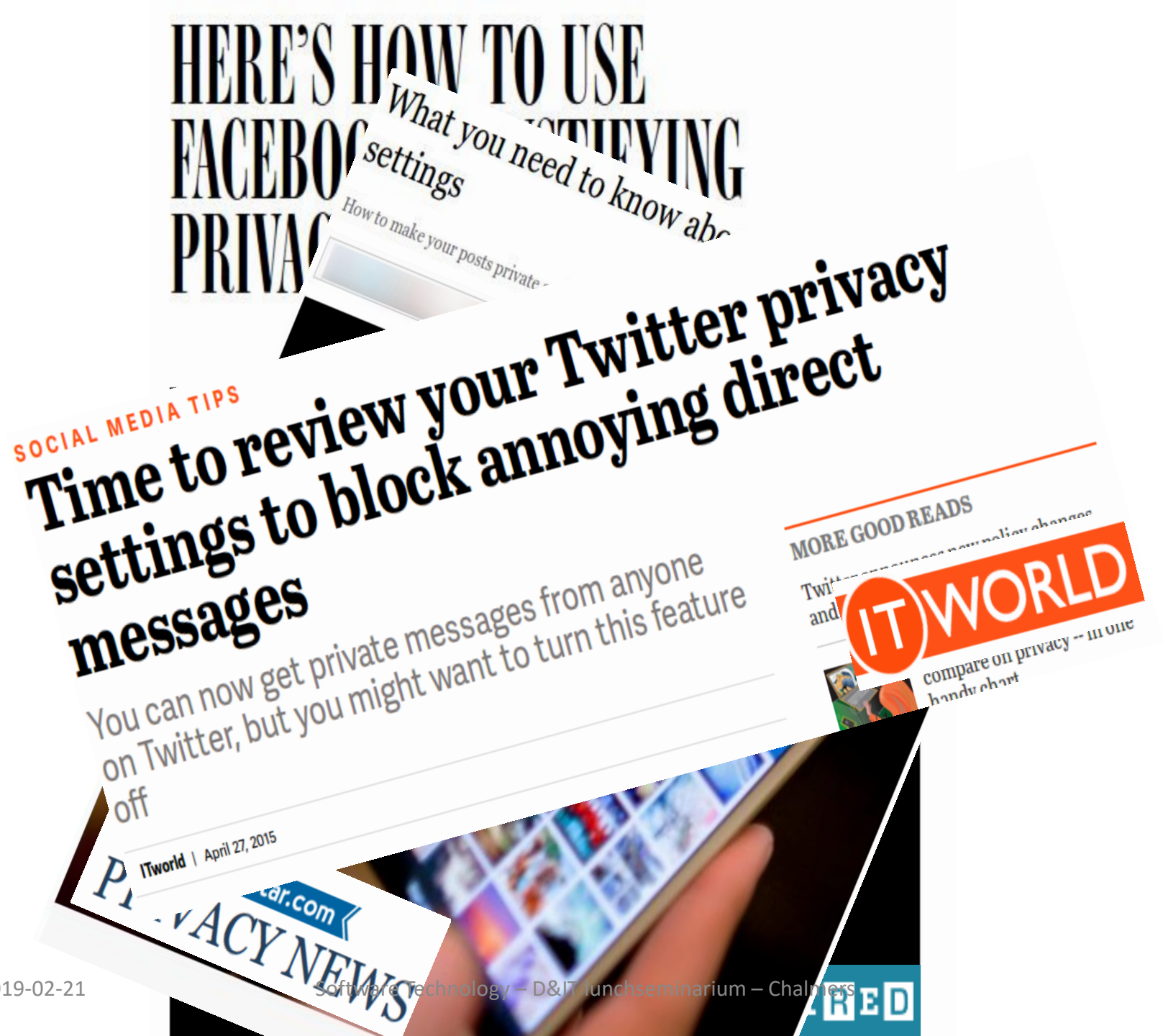
Like · Comment · Share

Devdatt and 20 people like this.

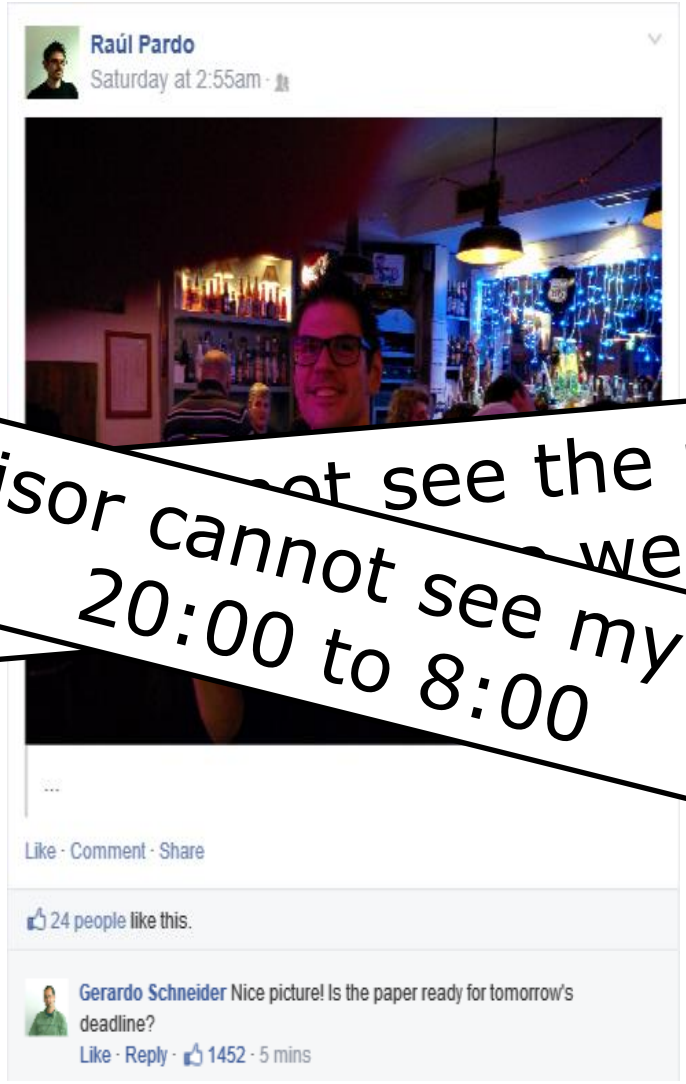
**Gerardo Schneider** Huh? Raúl is supposed to be working on tomorrow's presentation at FMPriv

Like · Reply · 15 · 5 mins

Write a comment ...


HERE'S HOW TO USE
What you need to know
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
MORE GOOD READS
IT WORLD
compare on privacy -- in one handy chart
PRIVACY NEWS
WIRED


# PRIVACY POLICIES




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

# Where to start?

TDA294 / DIT271

Formal Methods for Software Development, LP1

(DAT060 / DIT201 Logic in computer science, LP1)

(DAT350 / DIT232 Types for Programs and Proofs, LP1)

# All problems are not solved:


Information leakage due to speculation in hardware implementation.

2018: https://meltdownattack.com/