Construction and Verification of Software

2017 - 2018

MIEI - Integrated Master in Computer Science and Informatics
Consolidation block

Lecture 1 - Introduction and Motivation

João Costa Seco (joao.seco@fct.unl.pt)
based on previous editions by **Luís Caires** (lcaires@fct.unl.pt)

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Construction and Verification of Software

This course covers principles, methods, techniques and tools for the dependable and trustworthy construction and validation of software systems, ensuring as much as possible the absence of programming errors ("bugs"), with a focus on CONCURRENCY and SAFETY.

Project based learning using specialised techniques and tools.

Syllabus

- **Verified Software Construction**
  - Assertion methods and Hoare and Separation Logic; Assertion Inference; Abstract and Behavioural types. Representation Invariants. Abstract interpretation; Model-checking.
  - Hands On Exercises / Final Project using verification tools (Dafny, Verifast, INFER).
- **Software Testing**
  - Model-based testing; Test selection and test generation; Fault-based testing. Symbolic execution; Automated testing; Tools.
- **Concurrent Programming**
  - Sharing, confinement, ownership. Control of interference. Reasoning about concurrent code with monitors and locks based on resource invariants. Construction of concurrency control code from behavioral specs.

Objectives

- **Static Verification of Software**
  - understand the principles and know how to use assertion methods in practice to specify, reason about, and verify software
- **Dynamic Verification of Software**
  - understand the principles and methods for testing software.
- **ADTs and Concurrent Programming**
  - Write correct concurrent programs and ADTs
  - understand ADT programming methodologies
  - understand concurrent programming methodologies
  - understand ADT programming methodologies
  - understand concurrent programming methodologies

Bibliografia

Program Development In Java: Abstraction, Specification, and Object-Oriented Design.
Barbara Liskov (with John Guttag); MIT Press.

Code Complete: A Practical Handbook of Softwar
Construction, Second Edition.
Steve McConnell, Microsoft Press.

The Art of Software Testing, Second Edition
Glenford Myers, Corey Sandler, Tom Badgett

Java Concurrency in Practice,
Goetz et al. Addison-Wesley, 2006.

Tutorials for Dafny and Verifast

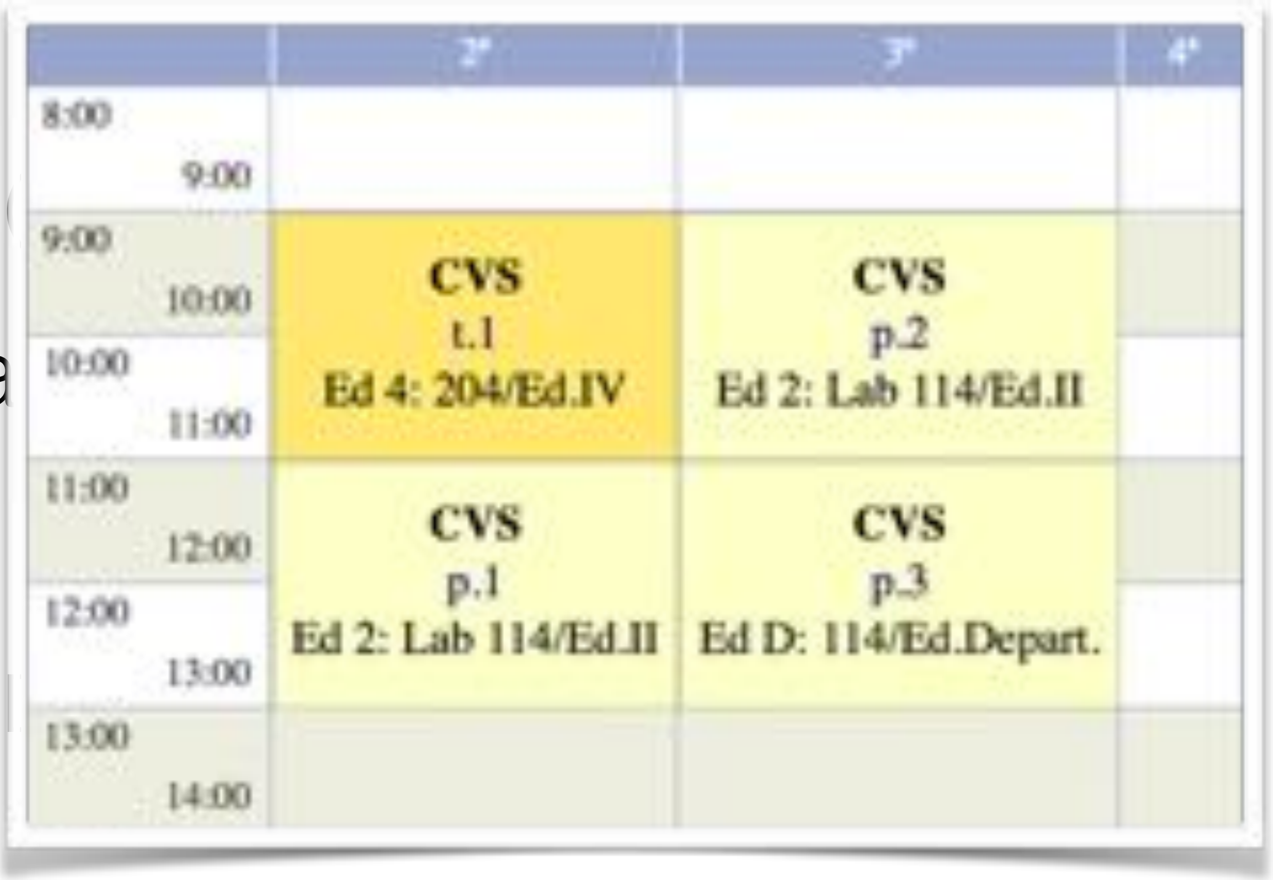
Program Development in Java
Abstraction, Specification, and Object-Oriented Design
Barbara Liskov
CODE COMPLETE 2
The ART of SOFTWARE TESTING
BRIAN GOETZ
JAVA CONCURRENCY IN PRACTICE

Logistics and Evaluation

- 13 Lectures
    - Midterm (w6) and Final Test (w12) — to be determined
- Lab Sessions (3-4 Lab classes) — to be determined
    - Teams of 1-2 students
    - Handouts - Test generation, Dafny exercises
    - Project (two deliveries)
      Development and verification of concurrent system - Verifast
- Communication: — to be determined soon
- Evaluation details not final in CLIP, will be updated

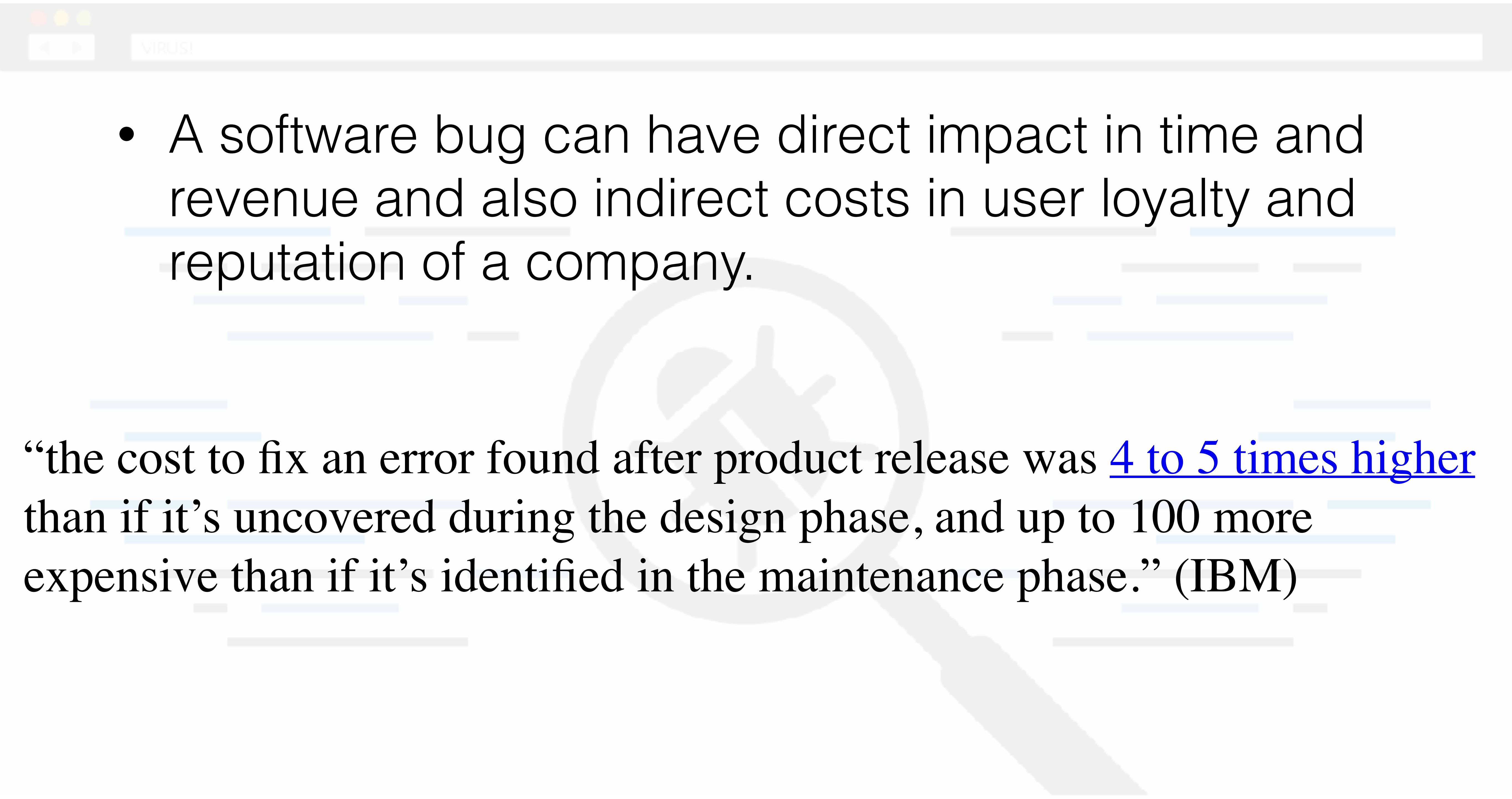
Logistics and Evaluation

- 13 Lectures
  - Midterm (w6) and Final Test
- Lab Sessions (3-4 Lab cla
  - Teams of 1-2 students
  - Handouts - Test generation,
  - Project (two deliveries)
    Development and verification of concurrent system - Verifast
- Communication: — to be determined soon
- Evaluation details not final in CLIP, will be updated



	2ª	3ª	4ª
8:00 9:00			
9:00 10:00	CVS t.1 Ed 4: 204/Ed.IV	CVS p.2 Ed 2: Lab 114/Ed.II	
10:00 11:00			
11:00 12:00	CVS p.1 Ed 2: Lab 114/Ed.II	CVS p.3 Ed D: 114/Ed.Depart.	
12:00 13:00			
13:00 14:00			

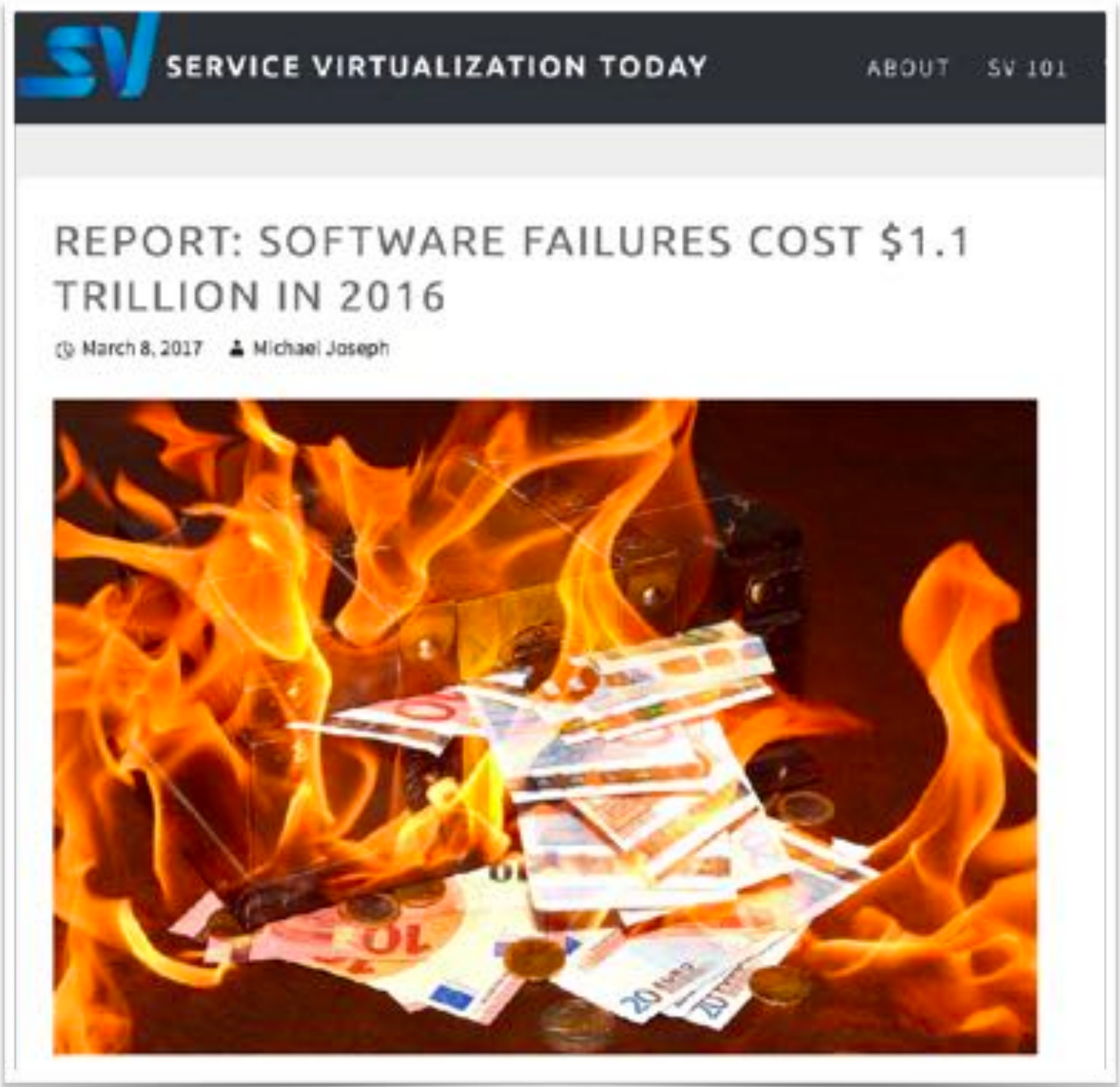
What’s the True Cost of a Software Bug?



- A software bug can have direct impact in time and revenue and also indirect costs in user loyalty and reputation of a company.

“the cost to fix an error found after product release was [4 to 5 times higher]() than if it’s uncovered during the design phase, and up to 100 more expensive than if it’s identified in the maintenance phase.” (IBM)

https://crossbrowsertesting.com/blog/development/software-bug-cost/



SERVICE VIRTUALIZATION TODAY

ABOUT SV 101

REPORT: SOFTWARE FAILURES COST $1.1 TRILLION IN 2016

March 8, 2017 Michael Joseph

Not really a new thing

- Byte Magazine 1995

SPECIAL!!! THE BYTE GUIDE TO GAMES
BYTE
Reviewed! Office 95, Symantec C++
Compared! 133 MHz Pentium Desktops
Inside the FBI's Database
THE MAGAZINE OF TECHNOLOGY INTEGRATION
HOW SOFTWARE DOESN'T WORK
AND WHAT YOU CAN DO ABOUT IT
PLUS
Artificial Intelligence Builds A Better Factory
Lessons from Air Traffic Control, page 48

Hardware bugs are even worse

- Byte Magazine March 1995

THE TRUTH BEHIND THE PENTIUM BUG

BYTE

THE MAGAZINE OF TECHNOLOGY INTEGRATION

New CD ThinkPad

Hofstadter on Artificial Intelligence

Tough New Cross-Platform Benchmarks

7 NEW WAYS TO LEARN

From boardroom to classroom—how advanced technology is reshaping the way we think.

PLUS

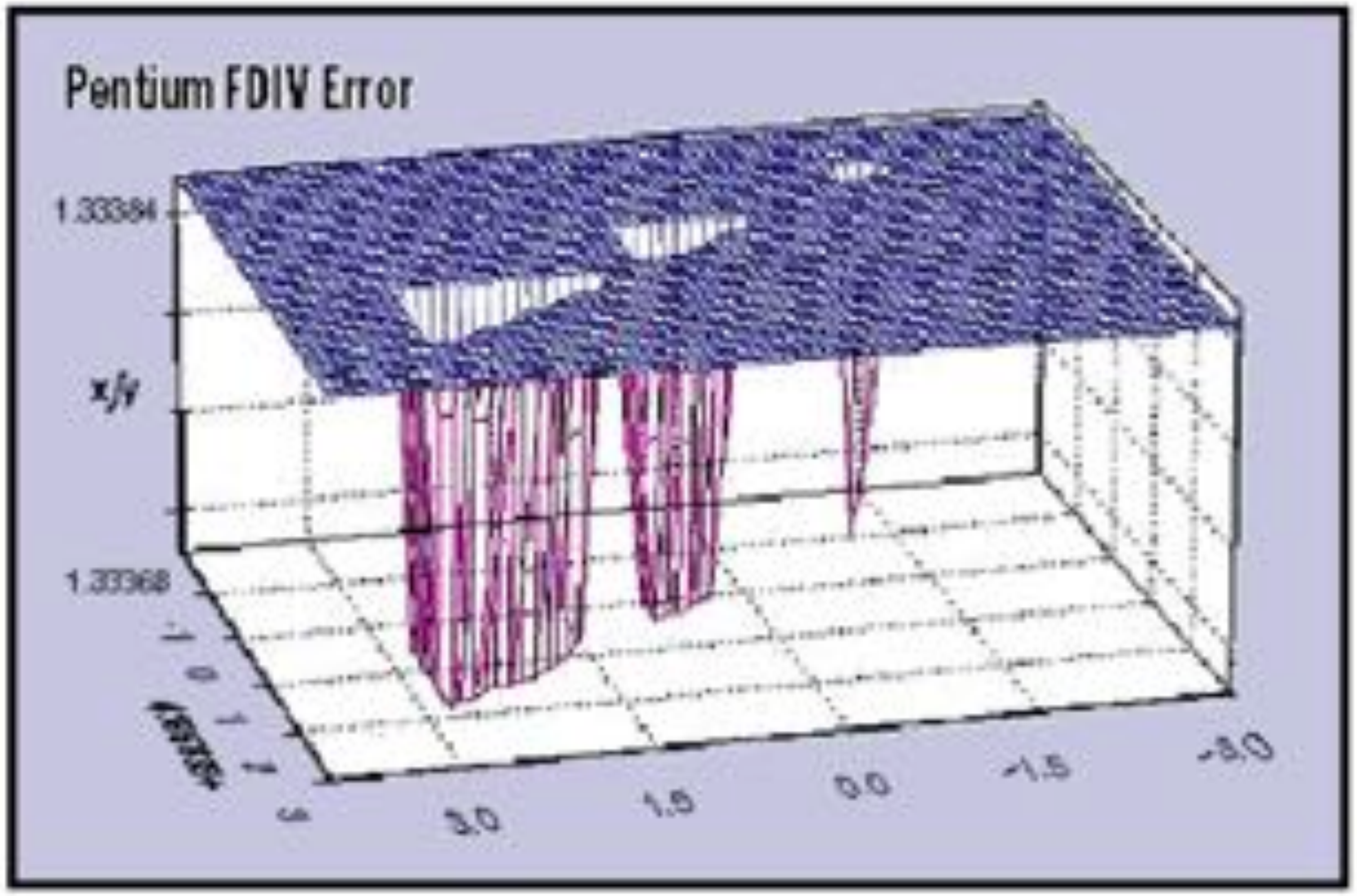
26 Tape Drives Tested

Is War [redacted]

App [redacted]

Fix PowerPC Standard

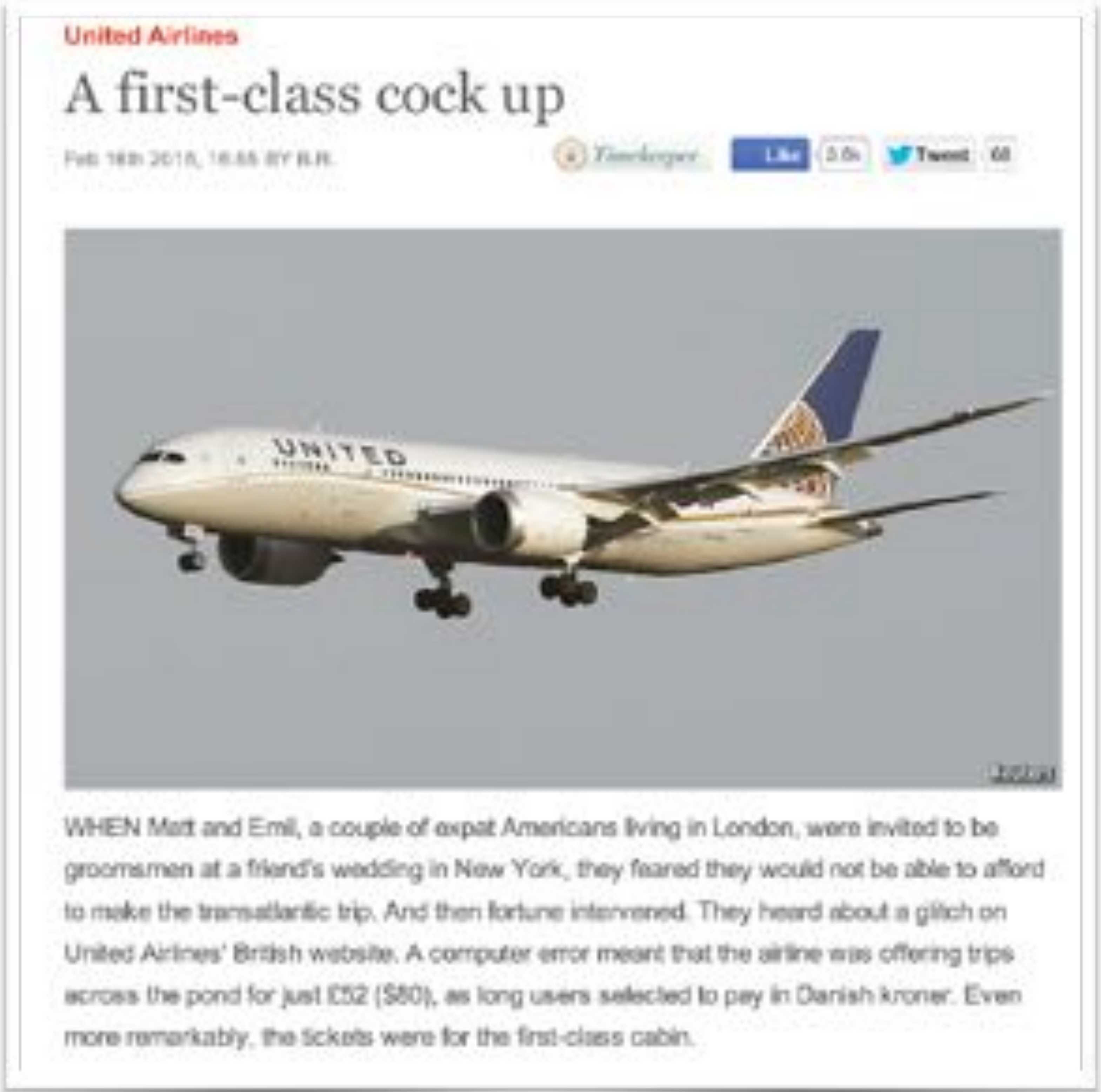
Hardware Bugs are even worse



A 3-D plot of the ratio 4195835/3145727 calculated on a Pentium with FDIV bug. The depressed triangular areas indicate where incorrect values have been computed. The correct values all would round to 1.3338, but the returned values are 1.3337, an error in the fifth significant digit. Byte Magazine, March 1995.

https://www.cs.earlham.edu/~dusko/cs63/fdiv.html

Too easy to make flawed software

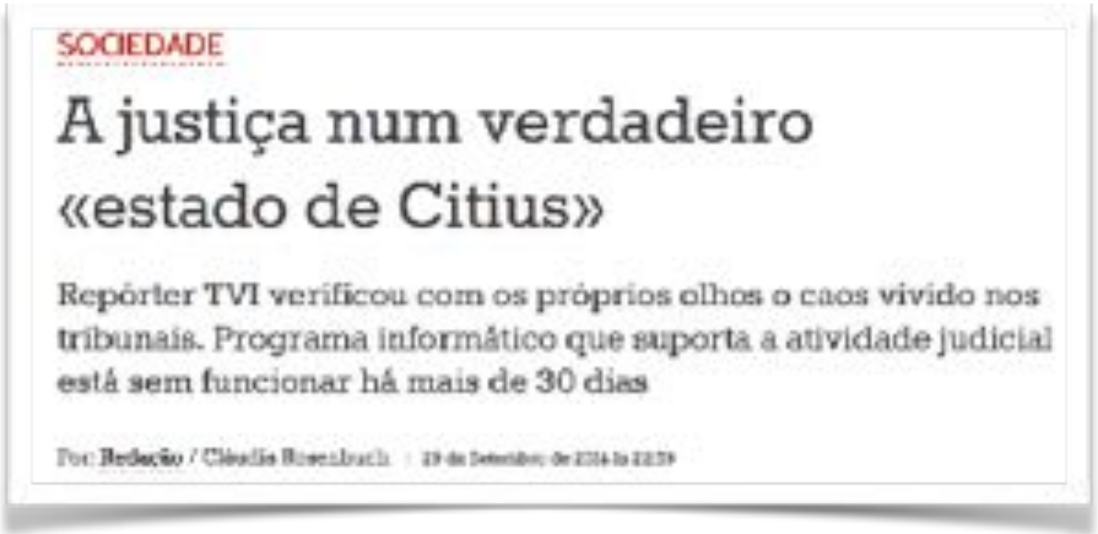


United Airlines

A first-class cock up

Feb 16th 2015, 16:55 BY B.R.

WHEN Matt and Emil, a couple of expat Americans living in London, were invited to be groomsmen at a friend's wedding in New York, they feared they would not be able to afford to make the transatlantic trip. And then fortune intervened. They heard about a glitch on United Airlines' British website. A computer error meant that the airline was offering trips across the pond for just £52 ($80), as long users selected to pay in Danish kroner. Even more remarkably, the tickets were for the first-class cabin.



SOCIEDADE

A justiça num verdadeiro «estado de Citius»

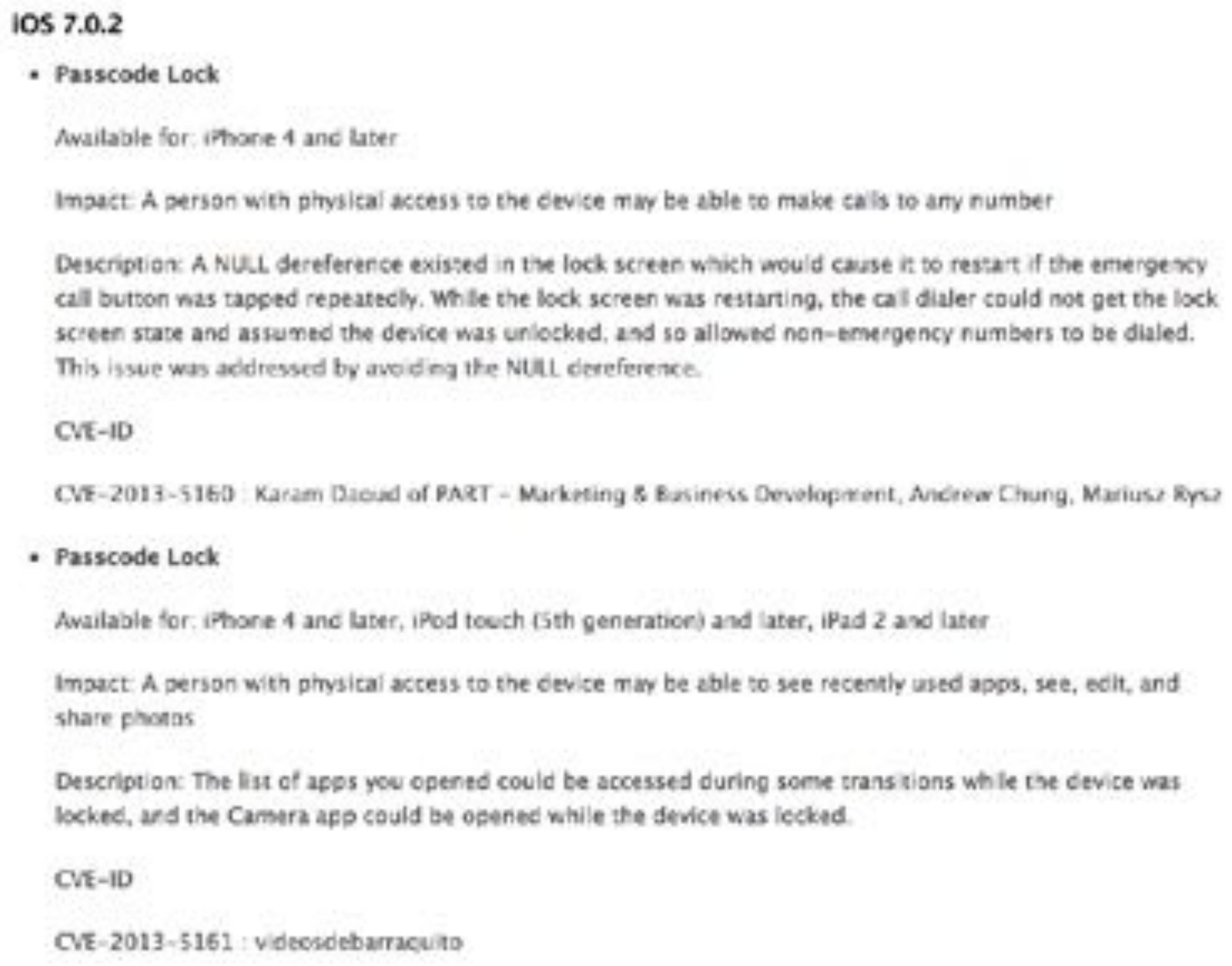
Repórter TVI verificou com os próprios olhos o caos vivido nos tribunais. Programa informático que suporta a atividade judicial está sem funcionar há mais de 30 dias

Por: Redação / Cláudia Rosenbusch | 29 de Setembro de 2014 às 22:39

Microsoft reveals Windows vulnerable to FREAK SSL flaw

[illegible]

Bug Report from Apple (2013)



iOS 7.0.2

- **Passcode Lock**

  Available for: iPhone 4 and later

  Impact: A person with physical access to the device may be able to make calls to any number

  Description: A NULL dereference existed in the lock screen which would cause it to restart if the emergency call button was tapped repeatedly. While the lock screen was restarting, the call dialer could not get the lock screen state and assumed the device was unlocked, and so allowed non-emergency numbers to be dialed. This issue was addressed by avoiding the NULL dereference.

  CVE-ID

  CVE-2013-5160 : Karam Daoud of PART - Marketing & Business Development, Andrew Chung, Mariusz Rysz

- **Passcode Lock**

  Available for: iPhone 4 and later, iPod touch (5th generation) and later, iPad 2 and later

  Impact: A person with physical access to the device may be able to see recently used apps, see, edit, and share photos

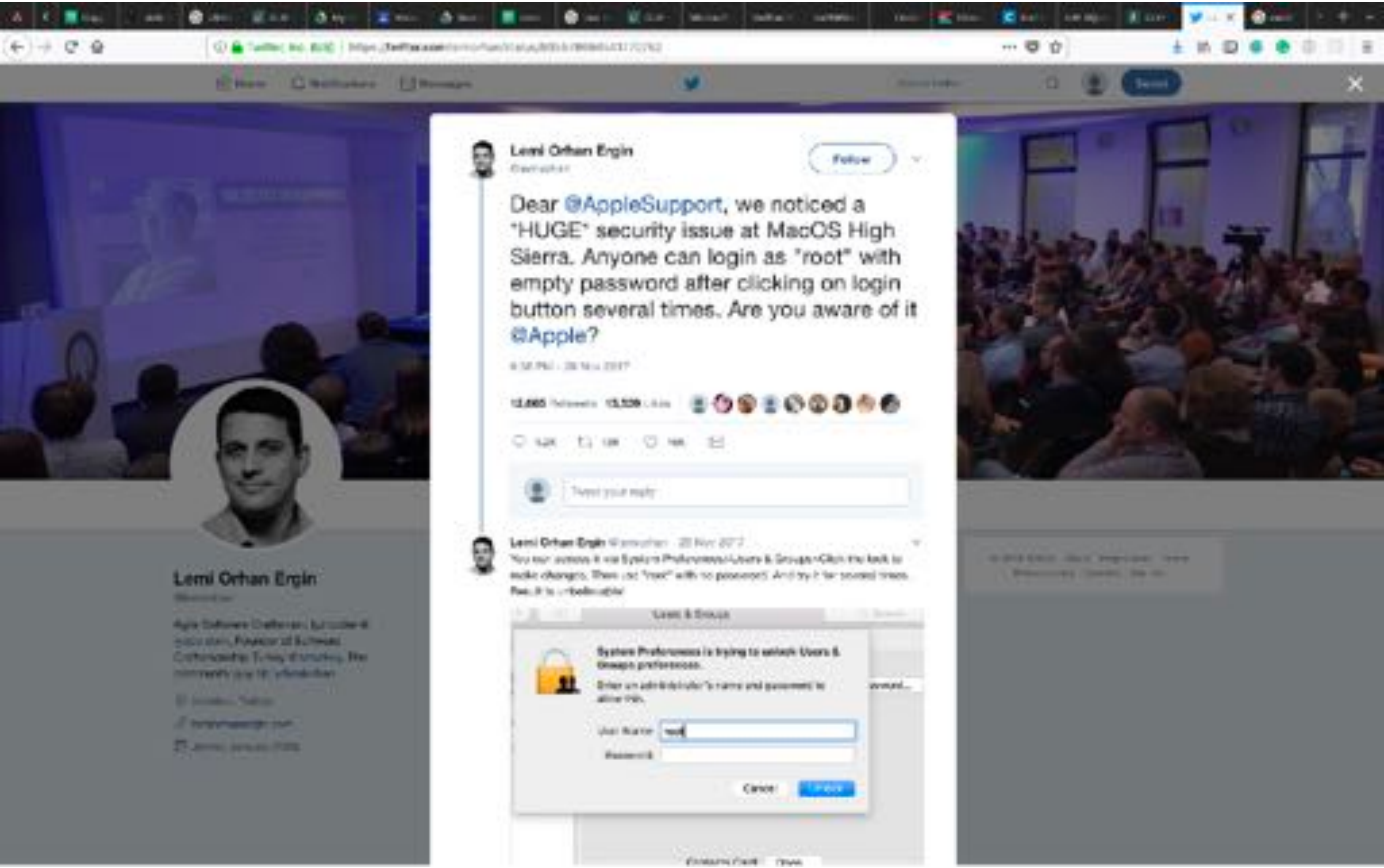
  Description: The list of apps you opened could be accessed during some transitions while the device was locked, and the Camera app could be opened while the device was locked.

  CVE-ID

  CVE-2013-5161 : videosdebarraquito

http://news.cnet.com/8301-1009_3-57603787-83/apple-promises-to-fix-ios-7-lock-screen-hack/

This is really bad!! (all over the news)



Lemi Orhan Ergin

Dear @AppleSupport, we noticed a *HUGE* security issue at MacOS High Sierra. Anyone can login as "root" with empty password after clicking on login button several times. Are you aware of it @Apple?

Lemi Orhan Ergin

You can access it via System Preferences>Users & Groups>Click the lock to make changes. Then use "root" with no password. And try it for several times. Result is unbelievable!

Users & Groups

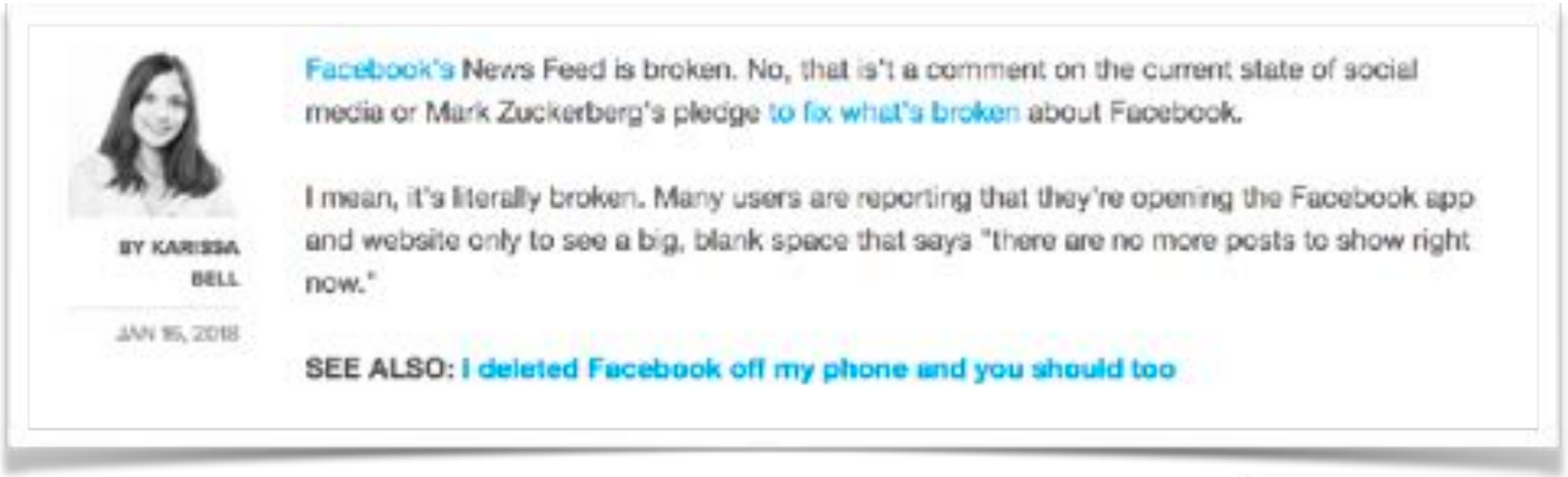
System Preferences is trying to unlock Users & Groups preferences.

Enter an administrator's name and password to allow this.

User Name: root

Password:

“Weird Facebook glitch breaks News Feed for some users”



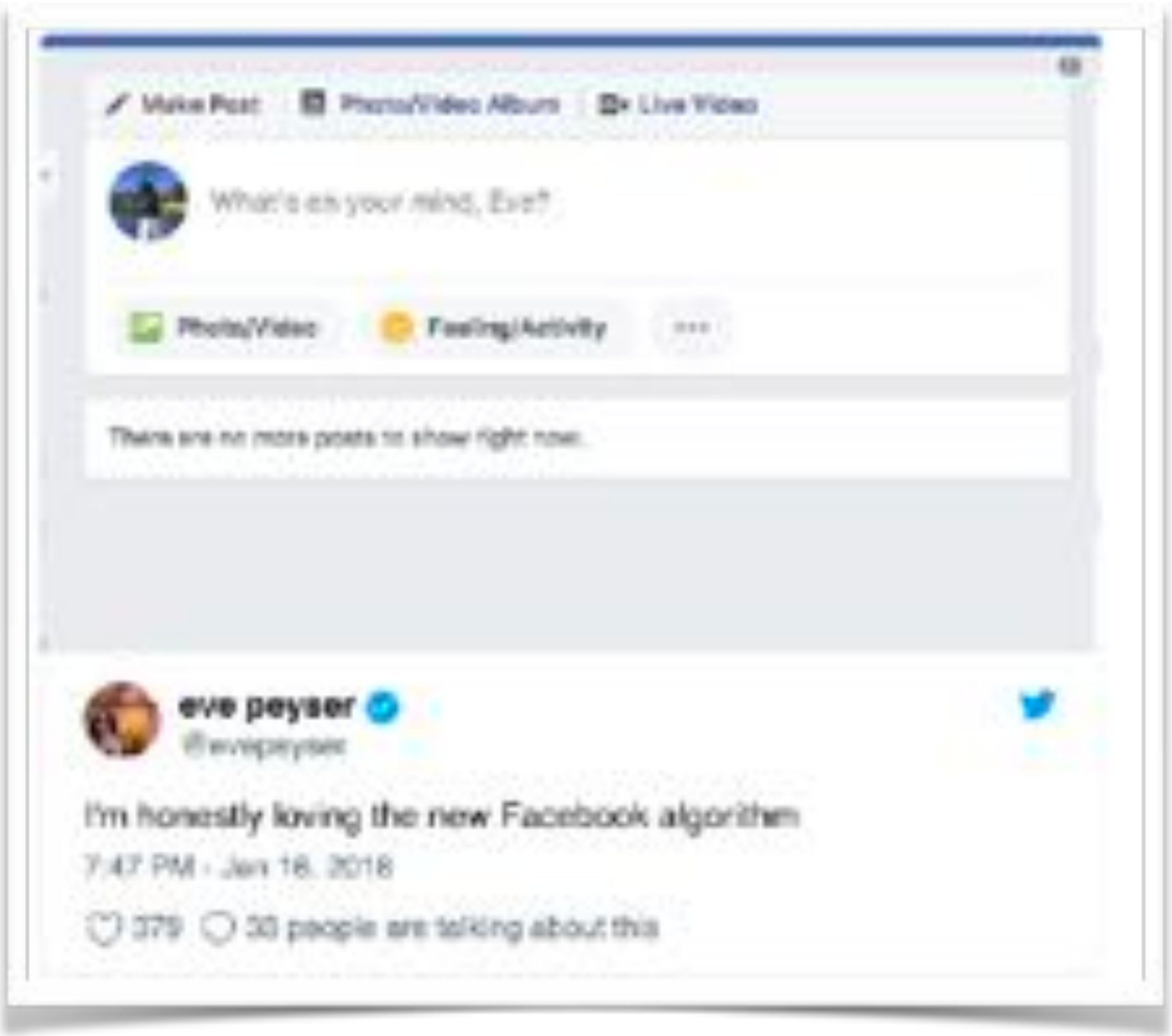
BY KARISSA BELL

JAN 16, 2018

Facebook's News Feed is broken. No, that isn't a comment on the current state of social media or Mark Zuckerberg's pledge to fix what's broken about Facebook.

I mean, it's literally broken. Many users are reporting that they're opening the Facebook app and website only to see a big, blank space that says "there are no more posts to show right now."

SEE ALSO: I deleted Facebook off my phone and you should too



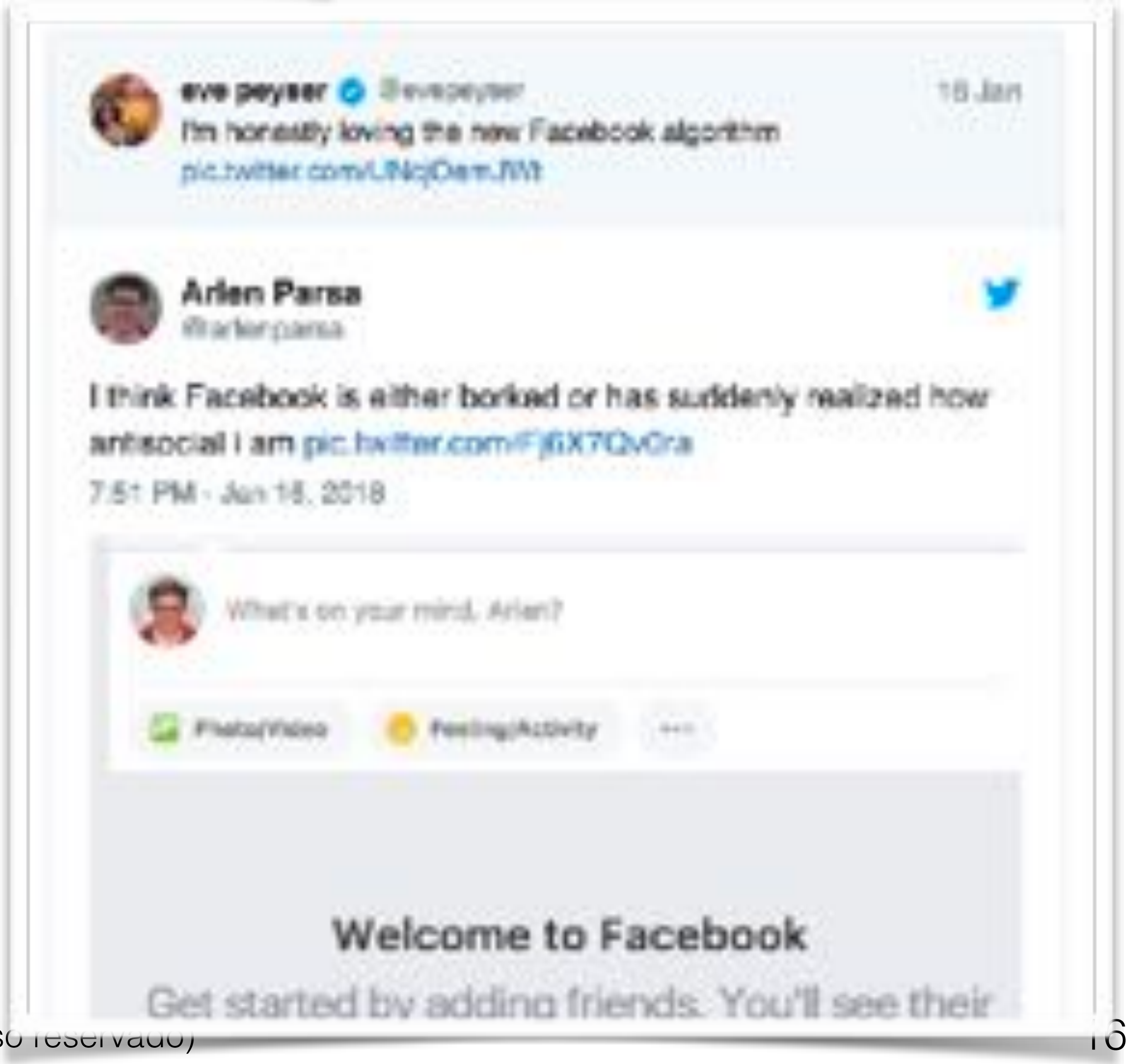
There are no more posts to show right now.

eve peyser @evepeyser

I'm honestly loving the new Facebook algorithm

7:47 PM - Jan 16, 2018

379 30 people are talking about this



eve peyser @evepeyser 16 Jan

I'm honestly loving the new Facebook algorithm pic.twitter.com/UNqDemJWt

Arlen Parsa @arlenparsa

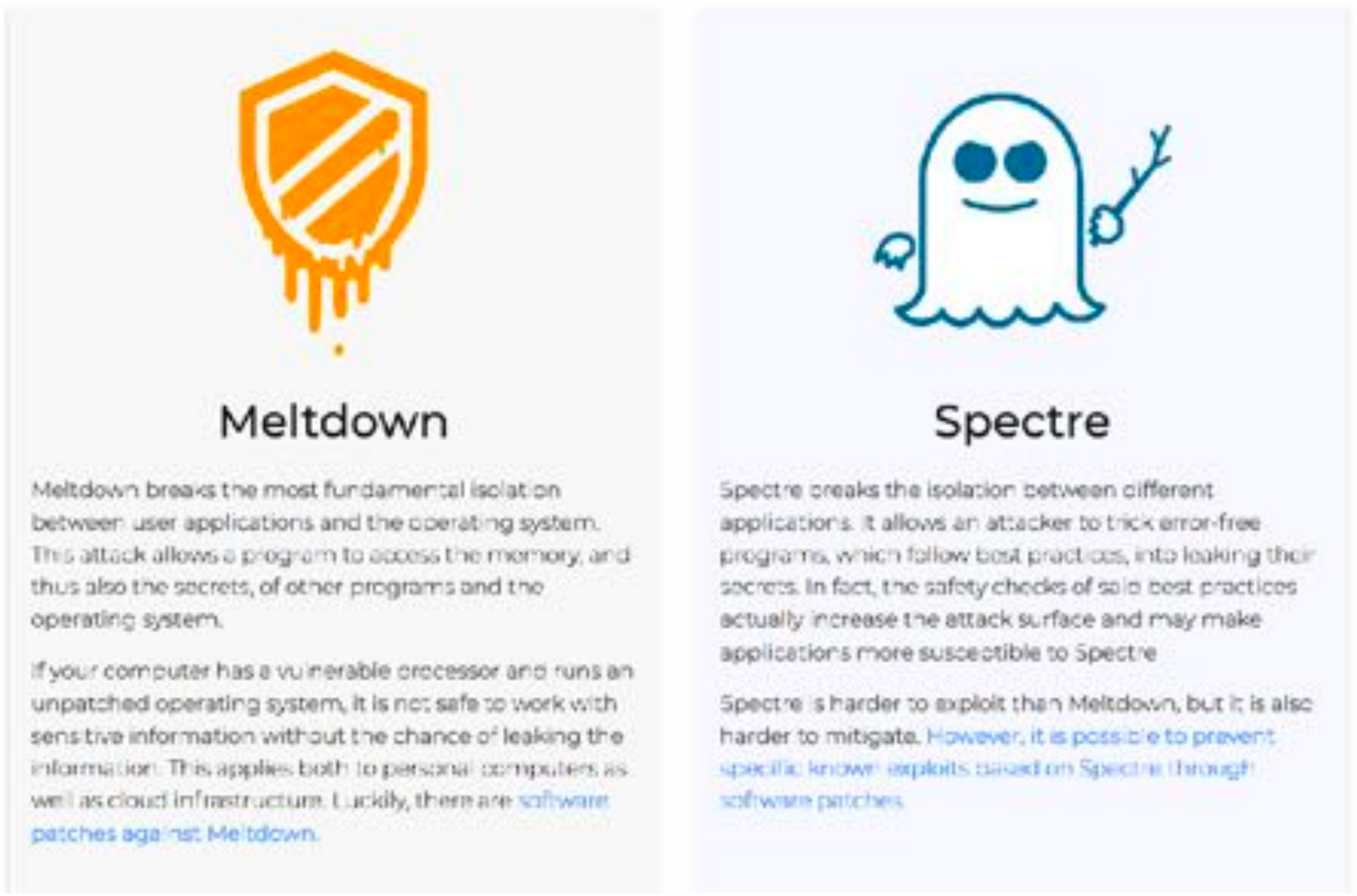
I think Facebook is either borked or has suddenly realized how antisocial I am pic.twitter.com/Fj6X7QvCra

7:51 PM - Jan 16, 2018

Welcome to Facebook

Get started by adding friends. You'll see their

https://meltdownattack.com/



Meltdown

Meltdown breaks the most fundamental isolation between user applications and the operating system. This attack allows a program to access the memory, and thus also the secrets, of other programs and the operating system.

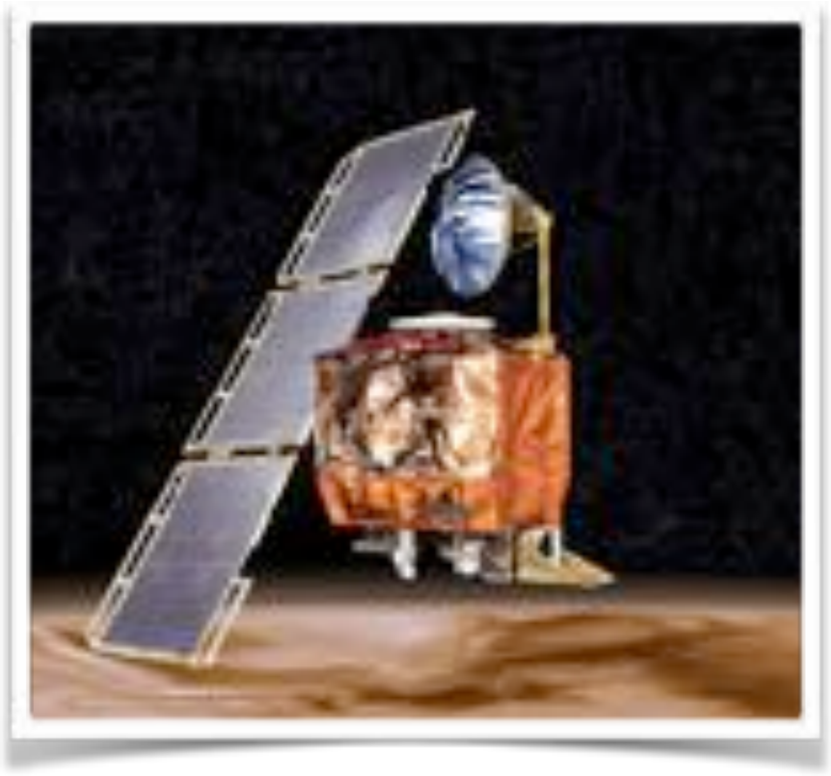
If your computer has a vulnerable processor and runs an unpatched operating system, it is not safe to work with sensitive information without the chance of leaking the information. This applies both to personal computers as well as cloud infrastructure. Luckily, there are software patches against Meltdown.

Spectre

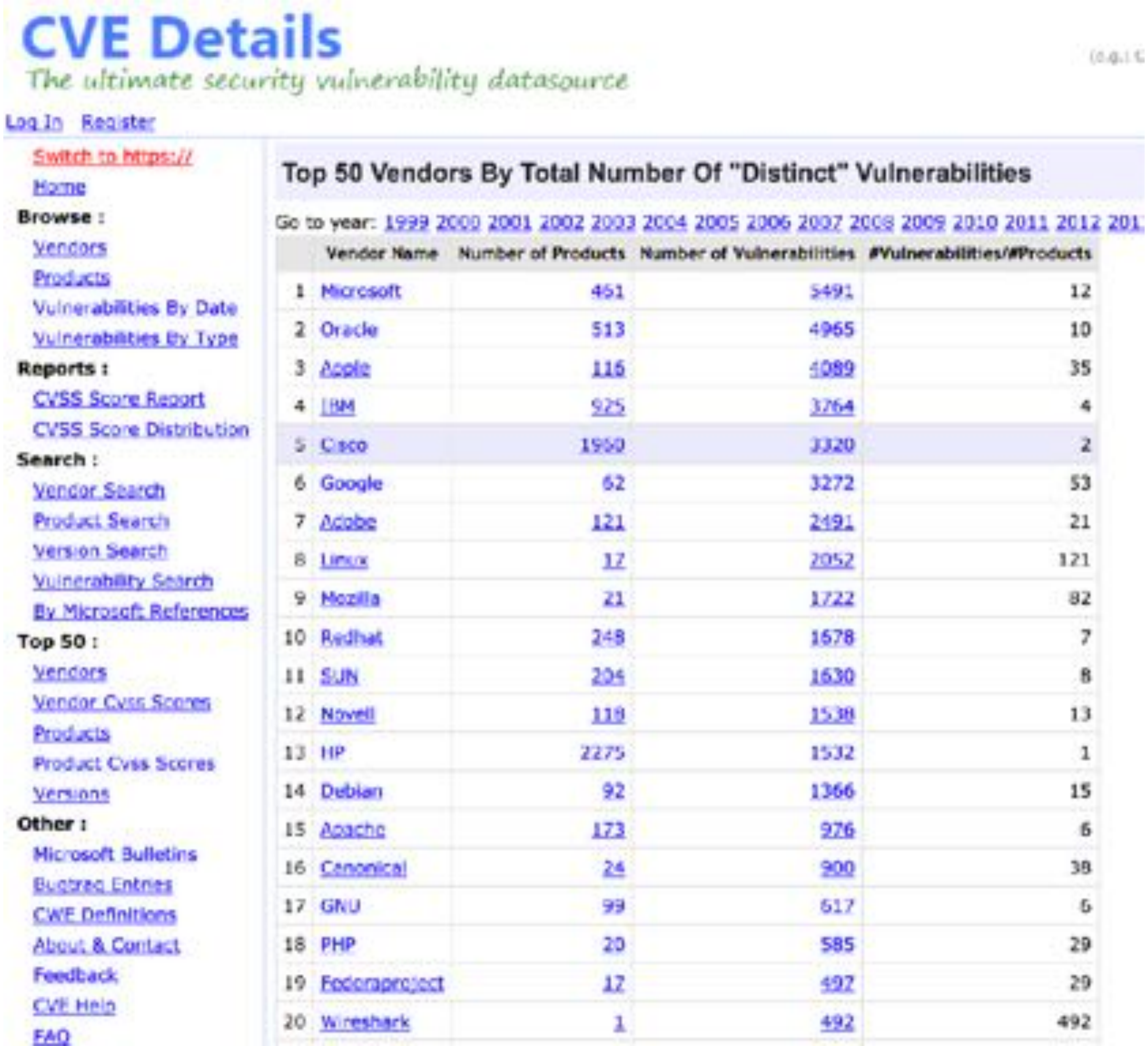
Spectre breaks the isolation between different applications. It allows an attacker to trick error-free programs, which follow best practices, into leaking their secrets. In fact, the safety checks of said best practices actually increase the attack surface and may make applications more susceptible to Spectre

Spectre is harder to exploit than Meltdown, but it is also harder to mitigate. However, it is possible to prevent specific known exploits based on Spectre through software patches.

Making Sure Software Really Works



- Software failures:
  - system crashes
  - unresponsive services
  - data losses
  - incorrect behaviours
  - security flaws
- can have huge <u>impacts</u>:
  - economic
    NASA's Mars Climate Orbiter - $125M+; Ariane5, $8B+;
  - user hassle
    FB - 2.2B; Gmail - 1B+; Instagram - 500M; Twitter - 330M; Netflix - 120M
  - data and systems <u>security</u>
    Vulnerabilities reported in 10y (Microsoft:3000, Oracle:3100, Apple:2600, …)
  - <u>military</u>
    Stuxnet (USA->Iran); F22 Crash; Patriot Missiles missed targets;



CVE Details

The ultimate security vulnerability datasource

Log In Register

Switch to https://
Home
Browse :
Vendors
Products
Vulnerabilities By Date
Vulnerabilities By Type
Reports :
CVSS Score Report
CVSS Score Distribution
Search :
Vendor Search
Product Search
Version Search
Vulnerability Search
By Microsoft References
Top 50 :
Vendors
Vendor Cvss Scores
Products
Product Cvss Scores
Versions
Other :
Microsoft Bulletins
Bugtraq Entries
CWE Definitions
About & Contact
Feedback
CVE Help
FAQ

Top 50 Vendors By Total Number Of "Distinct" Vulnerabilities

Go to year: 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 201

	Vendor Name	Number of Products	Number of Vulnerabilities	#Vulnerabilities/#Products
1	Microsoft	461	5491	12
2	Oracle	513	4965	10
3	Apple	116	4089	35
4	IBM	925	3764	4
5	Cisco	1950	3320	2
6	Google	62	3272	53
7	Adobe	121	2491	21
8	Linux	17	2052	121
9	Mozilla	21	1722	82
10	Redhat	248	1678	7
11	SUN	204	1630	8
12	Novell	119	1538	13
13	HP	2275	1532	1
14	Debian	92	1366	15
15	Apache	173	976	6
16	Canonical	24	900	38
17	GNU	99	617	6
18	PHP	20	585	29
19	Fedoraproject	17	497	29
20	Wireshark	1	492	492

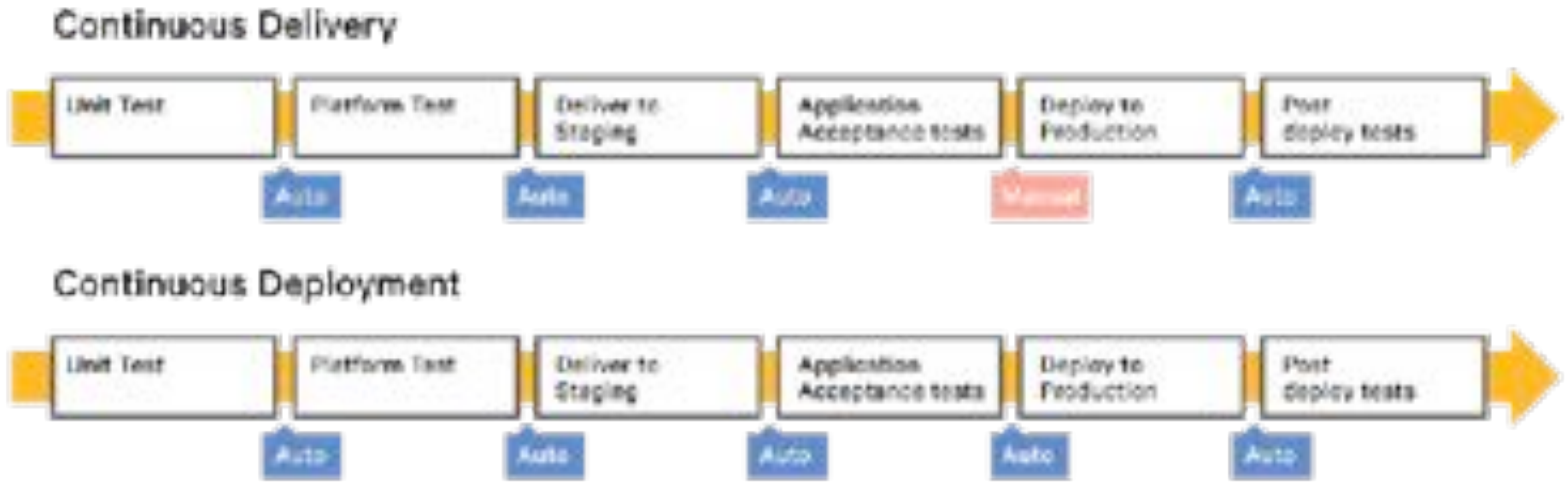
Pressure to update software fast

- Software development is increasingly competitive
- Any mistake can be extremely expensive
- Pressure is on to deliver fast and change even faster
- Companies deploy software at an astonishing pace:
  - Amazon: “every 11.7 seconds”
  - Netflix:
    “thousands of times per day”
  - Facebook:
    “bi-weekly app updates”

What’s the proper way of doing it?

Processes and Tools

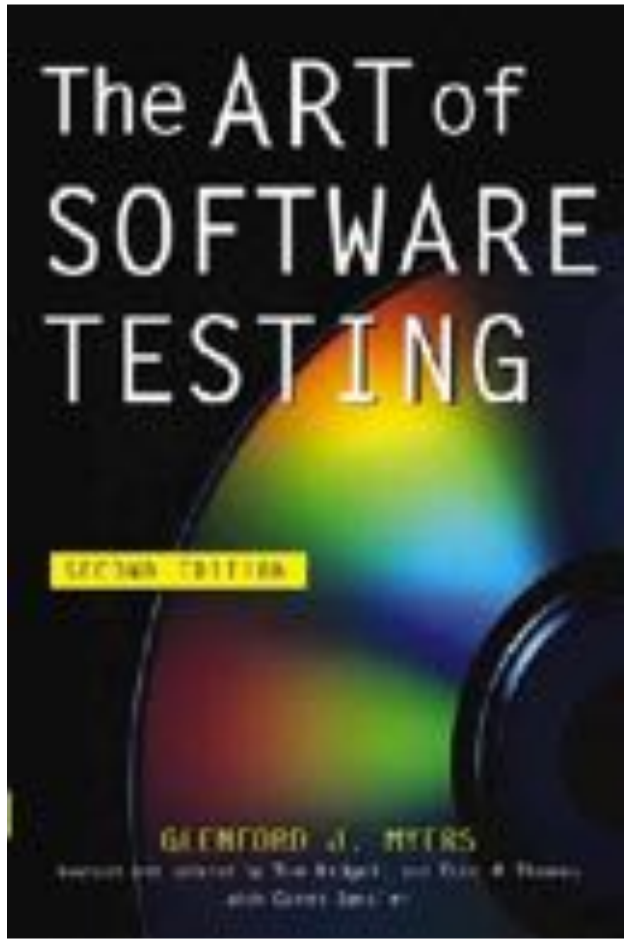
- Processes and Methods for software construction and software deployment (DevOps)
- Specification and development methods
- Testing tools and toolchains
- Validation and Verification techniques



Software Testing

“Software testing is a process, or a series of processes, designed to make sure computer code does what it was designed to do and that it does not do anything unintended”

The Art of Software Testing, Second Edition. Glenford Myers.



Software Testing - Validation

“Validation is the process designed to increase our confidence that a program works as intended. It can be done through verification or testing.”

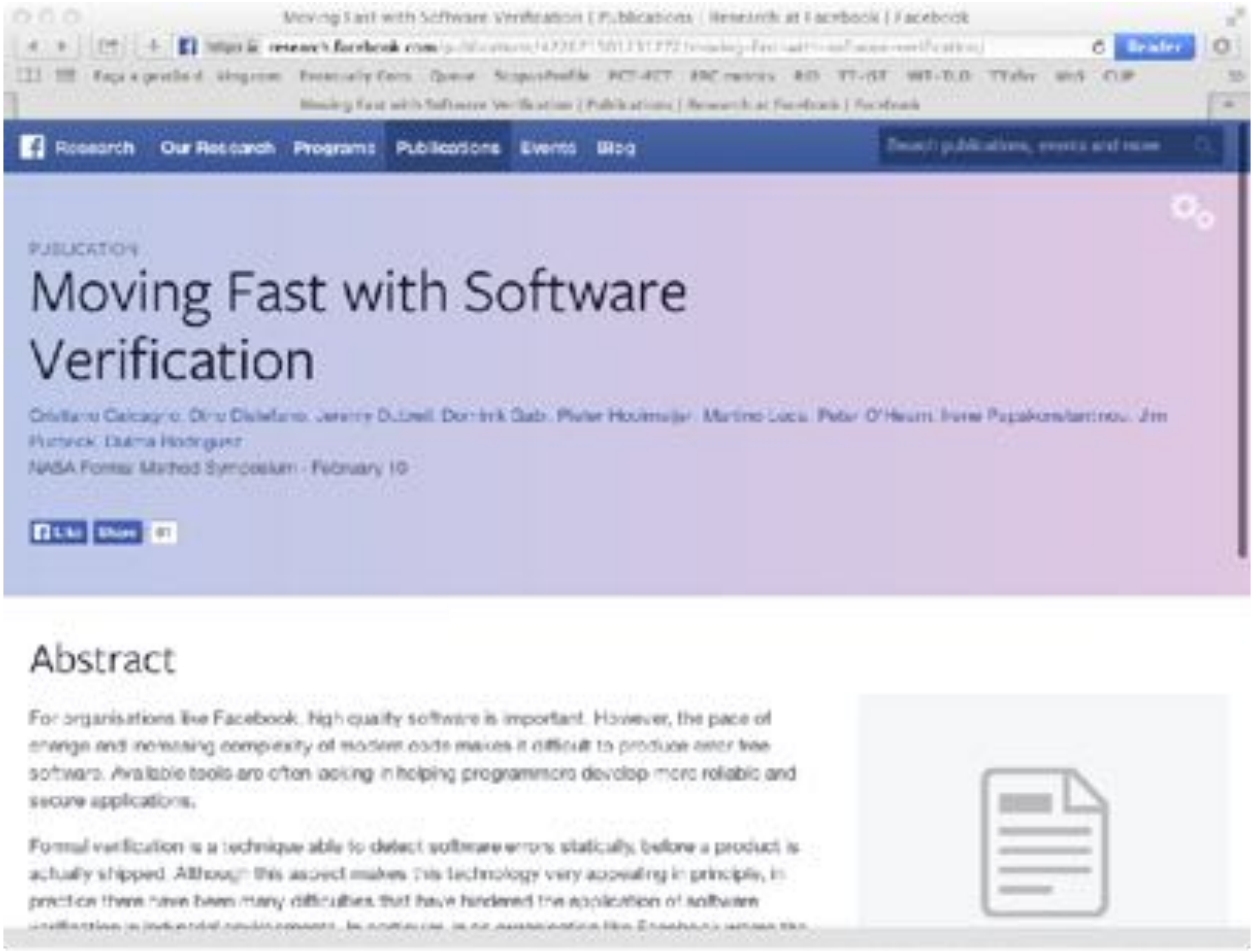
“Verification is a formal or informal argument that a program works on all possible inputs”.

“Testing is the process of running a program on a set of test cases and comparing the actual results with expected results”

in *Program Development in Java* (p222)

Program Development in Java
Abstraction, Specification, and Object-Oriented Design
Barbara Liskov
with John Guttag

Software Verification at Facebook


Moving Fast with Software Verification | Publications | Research at Facebook | Facebook
Research
Our Research
Programs
Publications
Events
Blog

PUBLICATION

Moving Fast with Software Verification

Cristiano Calcagno, Dino Distefano, Jeremy Dubreil, Dominik Gabi, Pieter Hooimeijer, Martino Luca, Peter O'Hearn, Irene Papakonstantinou, Jim Purbrick, Dulma Rodriguez

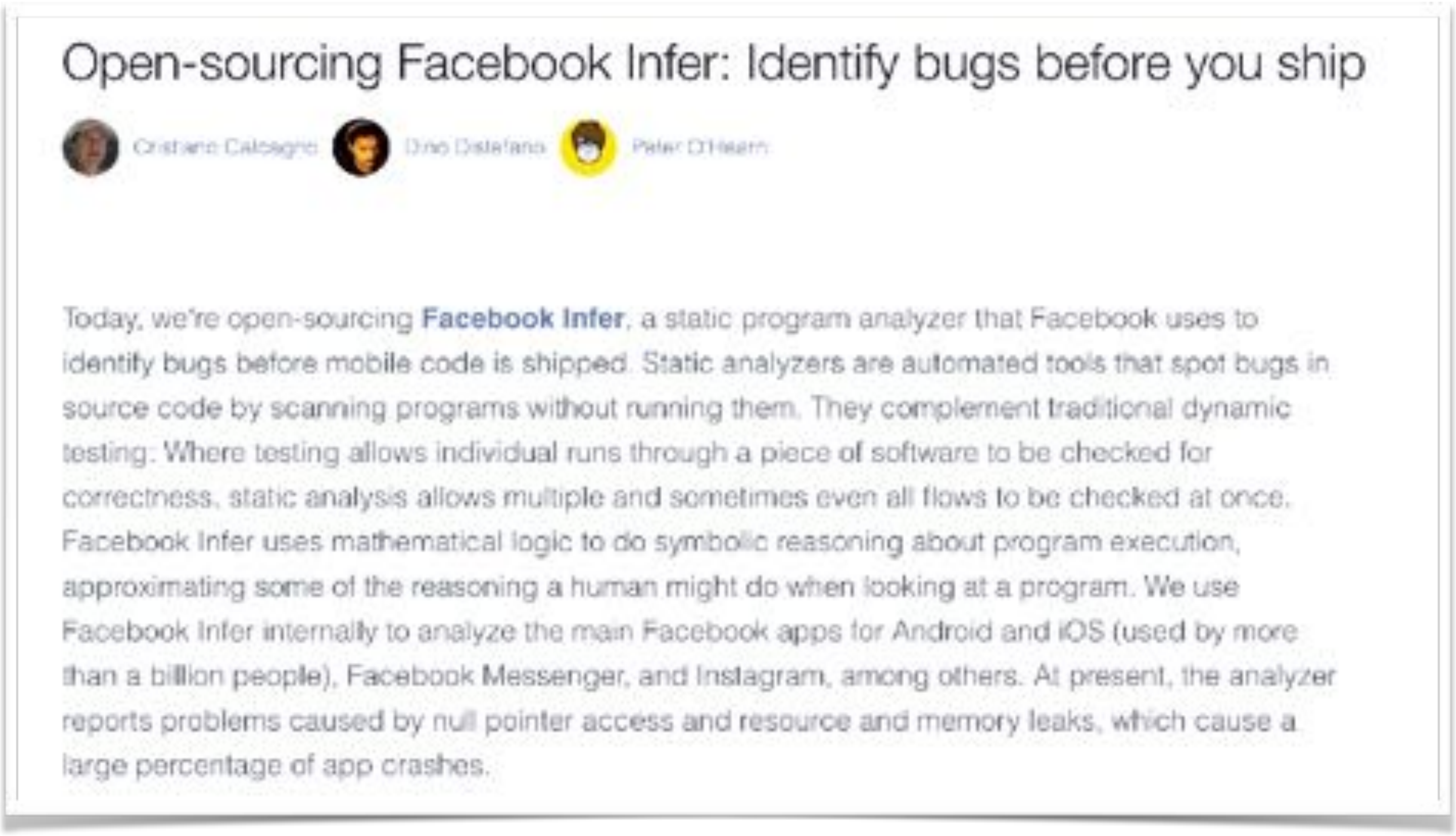
NASA Formal Method Symposium · February 19

Abstract

For organisations like Facebook, high quality software is important. However, the pace of change and increasing complexity of modern code makes it difficult to produce error free software. Available tools are often lacking in helping programmers develop more reliable and secure applications.

Formal verification is a technique able to detect software errors statically, before a product is actually shipped. Although this aspect makes this technology very appealing in principle, in practice there have been many difficulties that have hindered the application of software

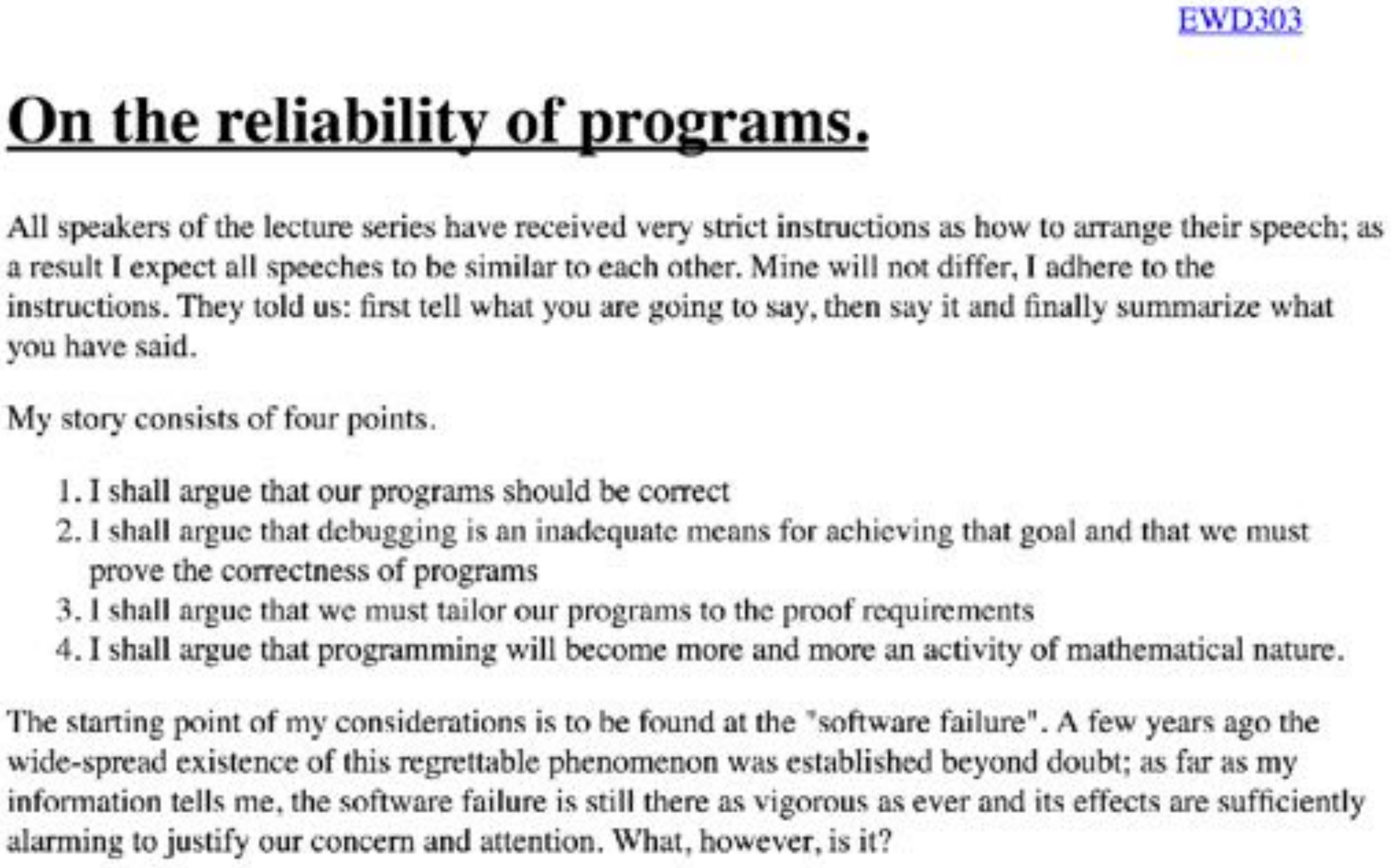
Software Verification at Facebook



Open-sourcing Facebook Infer: Identify bugs before you ship

Cristiano Calcagno Dino Distefano Peter O'Hearn

Today, we're open-sourcing **Facebook Infer**, a static program analyzer that Facebook uses to identify bugs before mobile code is shipped. Static analyzers are automated tools that spot bugs in source code by scanning programs without running them. They complement traditional dynamic testing: Where testing allows individual runs through a piece of software to be checked for correctness, static analysis allows multiple and sometimes even all flows to be checked at once. Facebook Infer uses mathematical logic to do symbolic reasoning about program execution, approximating some of the reasoning a human might do when looking at a program. We use Facebook Infer internally to analyze the main Facebook apps for Android and iOS (used by more than a billion people), Facebook Messenger, and Instagram, among others. At present, the analyzer reports problems caused by null pointer access and resource and memory leaks, which cause a large percentage of app crashes.



EWD303

On the reliability of programs.

All speakers of the lecture series have received very strict instructions as how to arrange their speech; as a result I expect all speeches to be similar to each other. Mine will not differ, I adhere to the instructions. They told us: first tell what you are going to say, then say it and finally summarize what you have said.

My story consists of four points.

1. I shall argue that our programs should be correct
2. I shall argue that debugging is an inadequate means for achieving that goal and that we must prove the correctness of programs
3. I shall argue that we must tailor our programs to the proof requirements
4. I shall argue that programming will become more and more an activity of mathematical nature.

The starting point of my considerations is to be found at the "software failure". A few years ago the wide-spread existence of this regrettable phenomenon was established beyond doubt; as far as my information tells me, the software failure is still there as vigorous as ever and its effects are sufficiently alarming to justify our concern and attention. What, however, is it?

https://www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD303.html

Testes de Software

“*Testing shows the presence, not the absence of bugs*”

Edsger W. Dijkstra, 1969

Software Correctness

Relevance of Software Correctness

- Quality procedures must be enforced at all levels, in particular at the construction phase, where most of the issues are introduced and difficult to circumvent.
- **Questions for you now**:
  - What methods do you currently use to make sure your code is “bullet-proof” ?
  - How can you prove to yourself (and others) that your code is “bullet-proof” ?
  - What arguments do you use to convince yourself and others that your code works as expected and not goes wrong, with respect to functional correctness, security, or concurrency errors?

Relevance of Software Correctness

- Quality procedures must be enforced at all levels, in particular at the construction phase, where most of the issues are introduced and difficult to circumvent.
- **Questions for you now**:
  - What methods do you currently use to make sure your code is "bullet-proof" ?
  - How can you prove to yourself (and others) that your code is "bullet-proof" ?
  - What arguments do you use to convince yourself and others that your code works as expected and not goes wrong, with respect to functional correctness, security, or concurrency errors?
- You will **know better answers** at the end of this course.

Software Correctness: What and How

- **Key engineering concern**:
  Make sure that the software developed and constructed is “correct”.
- What does this mean?
  - Is it crash-free? (“runtime safety”)
  - Gives the right results? (“functional correctness”)
  - Does it operate effectively? (“resource conformance”)
  - Does it violate user privacy? (“security conformance”)
  - …
- several process and methodological approaches to ensure and validate correctness exist (software engineering course)
- In this course, we cover some techniques to rigorously ensure and validate correctness **during software construction**

Software Correctness: What and How

- "runtime safety" (no crashes, etc.) is a bit easier to define
  - programming language type systems help a bit …
- other kinds of correctness are not so easy to define
- usually relative to special assumptions ...
  - what the system is supposed to do: play chess, manage bank accounts, …
  - the available resources: bandwidth, memory, processing speed, …
  - the security policies: only my friends can see my pics, …
- To precisely define such assumptions, we need
  - 1: precise specifications
  - 2: ways of validating that your system meets the spec

Correctness is against a specification

- Then what does “correct software” mean?
  - Always relative to some given (**our**) specs
- Correct means that software meets **our** specs
  - There is no such thing as the “right specification”
  - In practice, the spec is usually incomplete ...
  - But **the spec must not be wrong** !
  - It should be very easy to check what the spec states
  - The spec **must be simple, much simpler than code**
  - The spec should be **focused** (pick relevant cases)
    - e.g., buffers are not being overrun
    - e.g., never transfer money without logging the source

Checking Specs: Dynamic Verification

- By “dynamic verification” we mean that verification is **done at runtime**, while the program executes
- Some successful approaches:
    - **unit testing**
    - **coverage testing**
    - **regression testing**
    - **test generation**
    - **runtime monitoring**
- use runtime monitors to (continuously) check that code do not violate correctness properties
- violations causes exceptional behaviour or halt, so errors are detected after something wrong already occurred (think of a car crash, or a securiy leak)

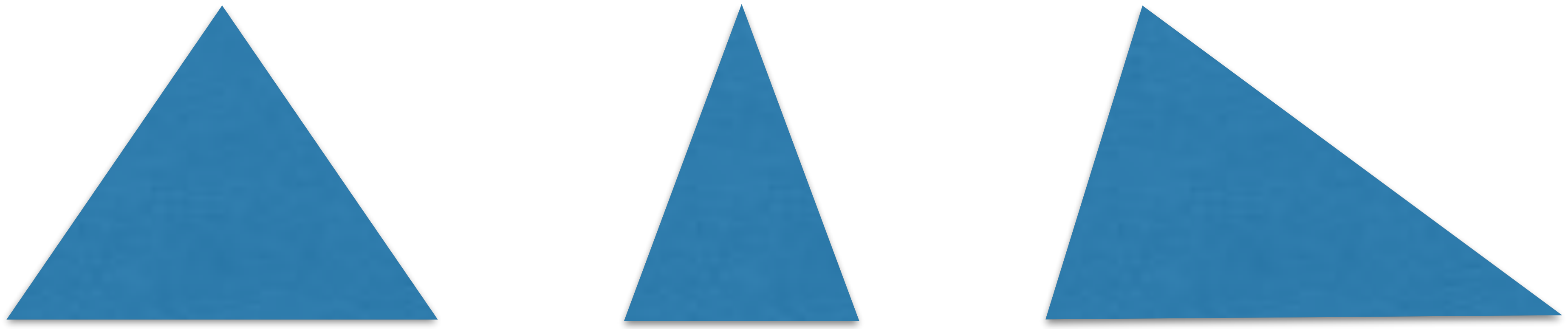
Checking Specs: Dynamic Verification

- Some shortcomings of dynamic verification
  - always introduces a level of performance overhead
  - may show the existence of some errors, but does not ensure absence of errors (the code passed a test suite today, but may fail with some other clever test)
- **Challenge**: how do you make sure that you are defining the “right” tests and “enough” tests
- Will talk about testing methods later on in the course

Quiz

Vamos ver quem sabe testar…

“The program reads three integer values from an input dialog. The three values represent the lengths of the sides of a triangle. The program displays a message that states whether the triangle is scalene, isosceles, or equilateral.”



Create specific tests (10 minutes)

Quiz

1. Do you have a test case that represents a valid scalene triangle? (Cases such as 1, 2, 3 and 2, 5, 10 are not valid triangles)
2. Do you have a test case that represents a valid equilateral triangle?
3. Do you have a test case that represents a valid isosceles triangle? (Cases such as 2,2,4 are not valid triangles.)
4. Do you have at least three test cases that represent valid isosceles triangles such that you have tried all three permutations of two equal sides (e.g. 3,3,4; 3,4,3; and 4,3,3)?
5. Do you have a test case in which one side has a zero value?
6. Do you have a test case in which one side has a negative value?

Quiz

7. Do you have a test case with three integers greater than zero such that the sum of two of the numbers is equal to the third? (If 1,2,3 is a scalene triangle, it's a bug.)

8. Do you have at least three test cases in category 7 such that you have tried all three permutations where the length of one side is equal to the sum of the lengths of the other two sides (for example, 1,2,3; 1,3,2; and 3,1,2)?

9. Do you have a test case with three integers greater than zero such that the sum of two of the numbers is less than the third (such as 1,2,4 or 12,15,30)?

10. Do you have at least three test cases in category 9 such that you have tried all three permutations (for example, 1,2,4; 1,4,2; and 4,1,2)?

Quiz

11. Do you have a test case in which all sides are zero (0,0,0)?
12. Do you have at least one test case specifying noninteger values (such as 2.5,3.5,5.5)?
13. Do you have at least one test case specifying the wrong number of values (two rather than three integers, for example)?
14. For each test case did you specify the expected output from the program in addition to the input values?

resultado = ?

Quiz

11. Do you have a test case in which all sides are zero (0,0,0)?

12. Do you have at least one test case specifying noninteger

13. Do you ha the wrong number of rs, for example)?

14. For each t ed output from the p s?

0 to 4	🟥
5 to 7	🟩
8+	

Readings

- Cost of Bugs
  https://crossbrowsertesting.com/blog/development/software-bug-cost/
- Pentium Bug 1990s
  https://www.cs.earlham.edu/~dusko/cs63/fdiv.html
- Meltdown and Spectre
  https://meltdownattack.com/
- EWD303
  https://www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD303.html
- EWD268 Structured Programming
  https://www.cs.utexas.edu/users/EWD/transcriptions/EWD02xx/EWD268.html
- Program Development in Java, Liskov/Guttag (ch1 and ch10).
- “Dafny: An Automatic Program Verifier for Functional Correctness”, Leino.