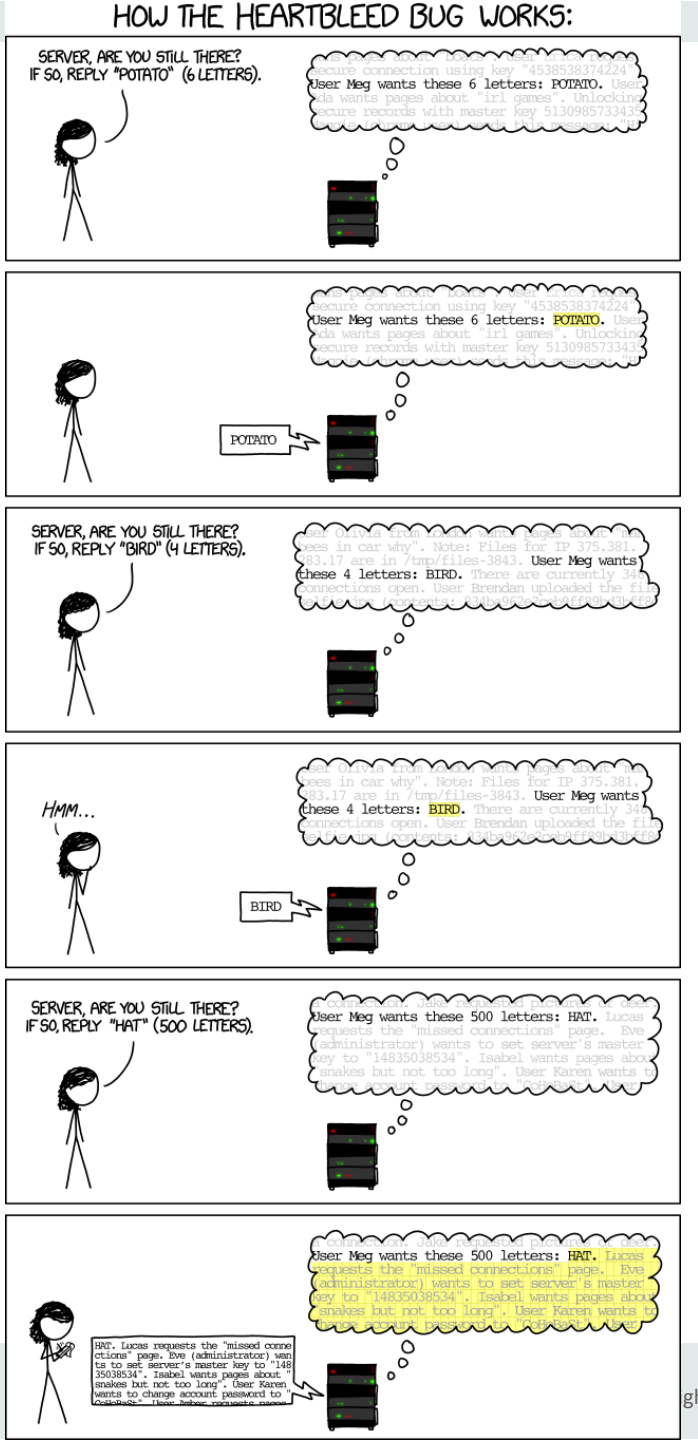
Are We Ready For Secure Languages?

Cristina Cifuentes
Chief Galah
Oracle Labs Australia
July 2016

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



https://xkcd.com/1354/

ORACLE

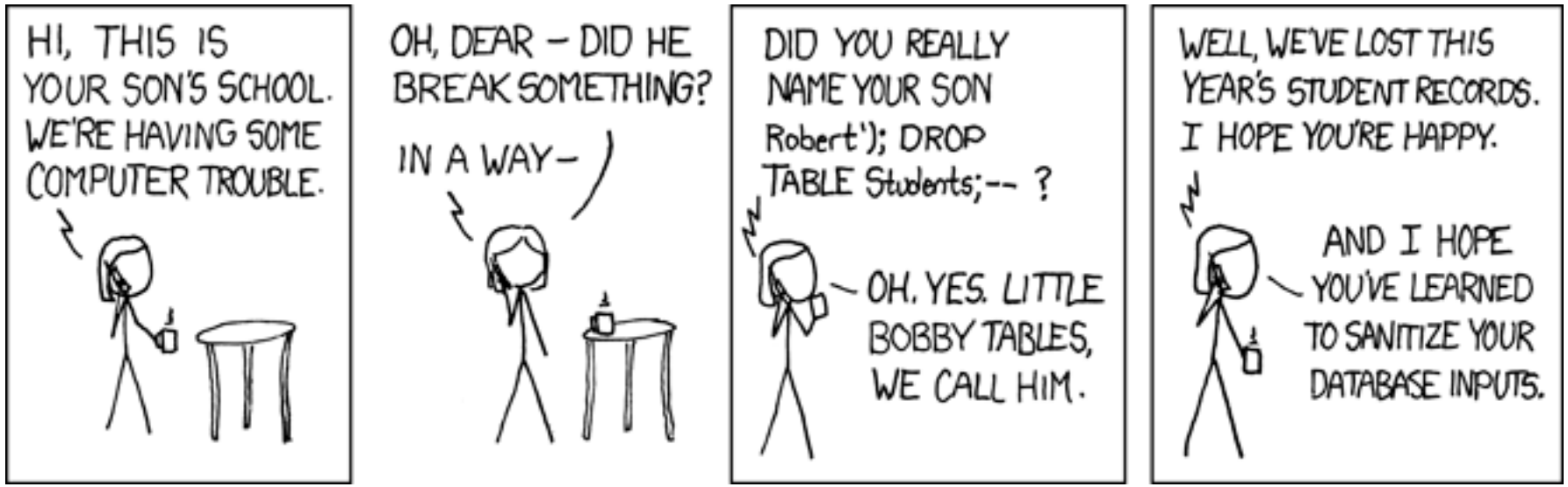
ghts reserved. |

Buffer errors

https://xkcd.com/1354/

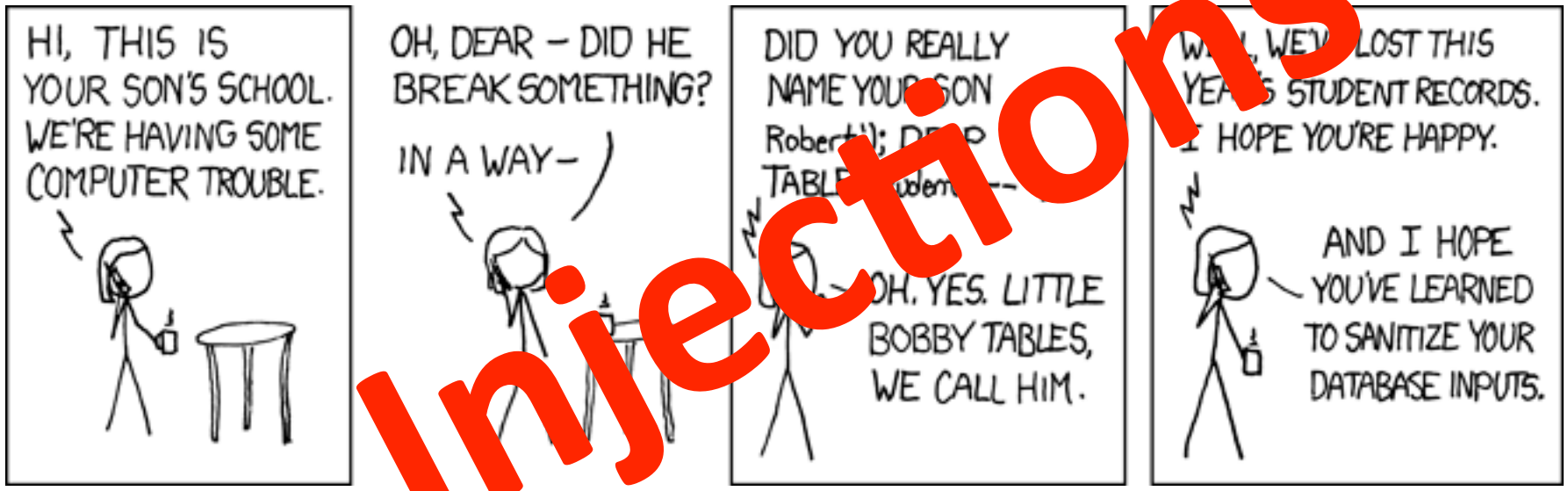
ORACLE

ghts reserved. |



https://xkcd.com/327/

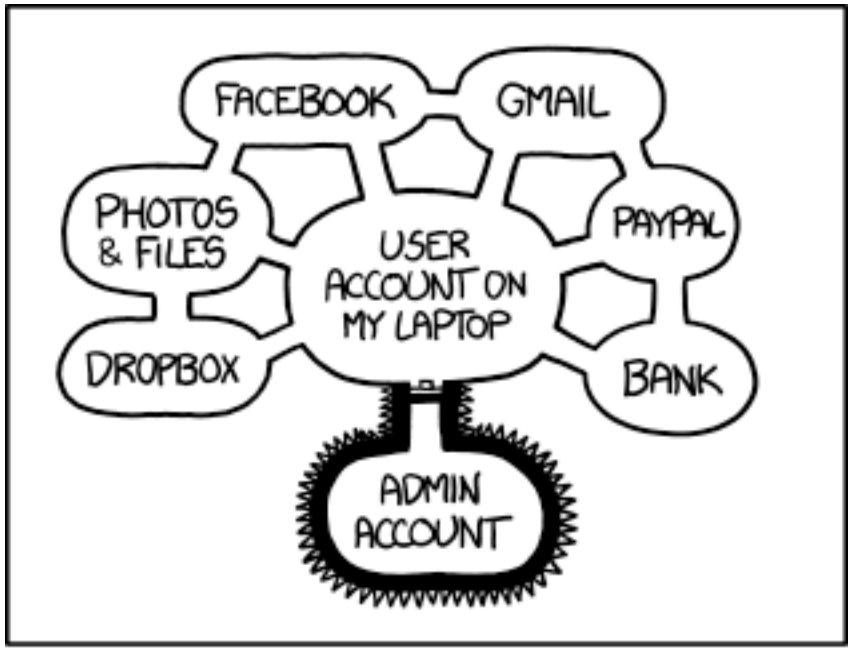
Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



Injections

https://xkcd.com/327/

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

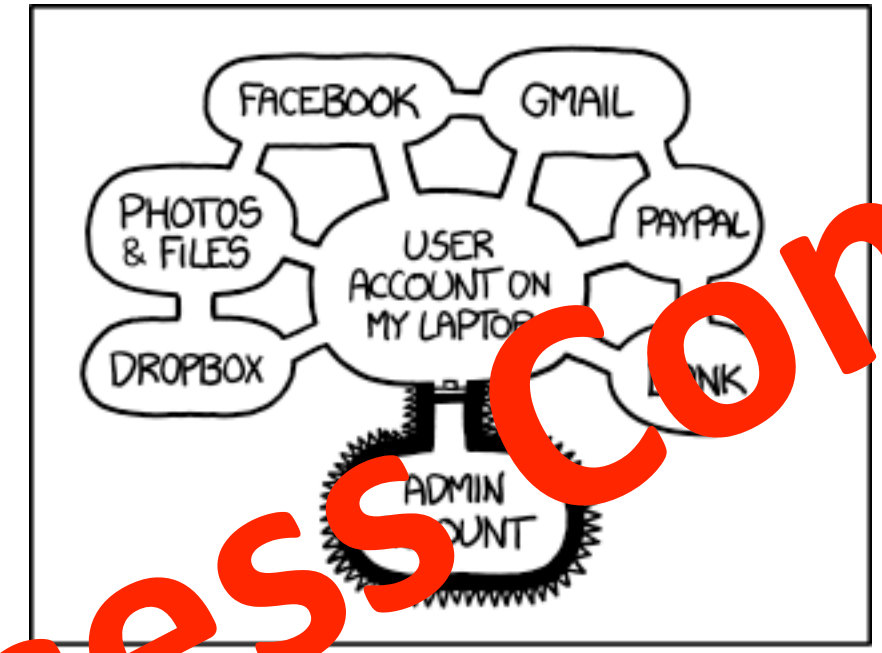


IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS,

BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION.

https://www.xkcd.com/1200/

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS,

BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION.

Access Control

https://www.xkcd.com/1200/

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

C, C++

Java
Java <-> C

Java EE
Java <-> C, Java <-> PL/SQL
JavaScript

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. | Confidential – Oracle Internal

2477

Vulnerabilities due to **buffer errors** (2013-2015)

National Vulnerability Database, http://nvd.nist.gov

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

2230

Vulnerabilities due to **cross-site scripting** (2013-2015)

National Vulnerability Database, http://nvd.nist.gov

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

1793

Vulnerabilities due to **permissions, privileges and access control** (2013-2015)

National Vulnerability Database, http://nvd.nist.gov

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

1972 Buffer overflow used in a kernel [1]

1988 Buffer overflow used in the Morris worm

1990s Cross-site scripting exploits

1998 SQL injection explained in the literature [2]

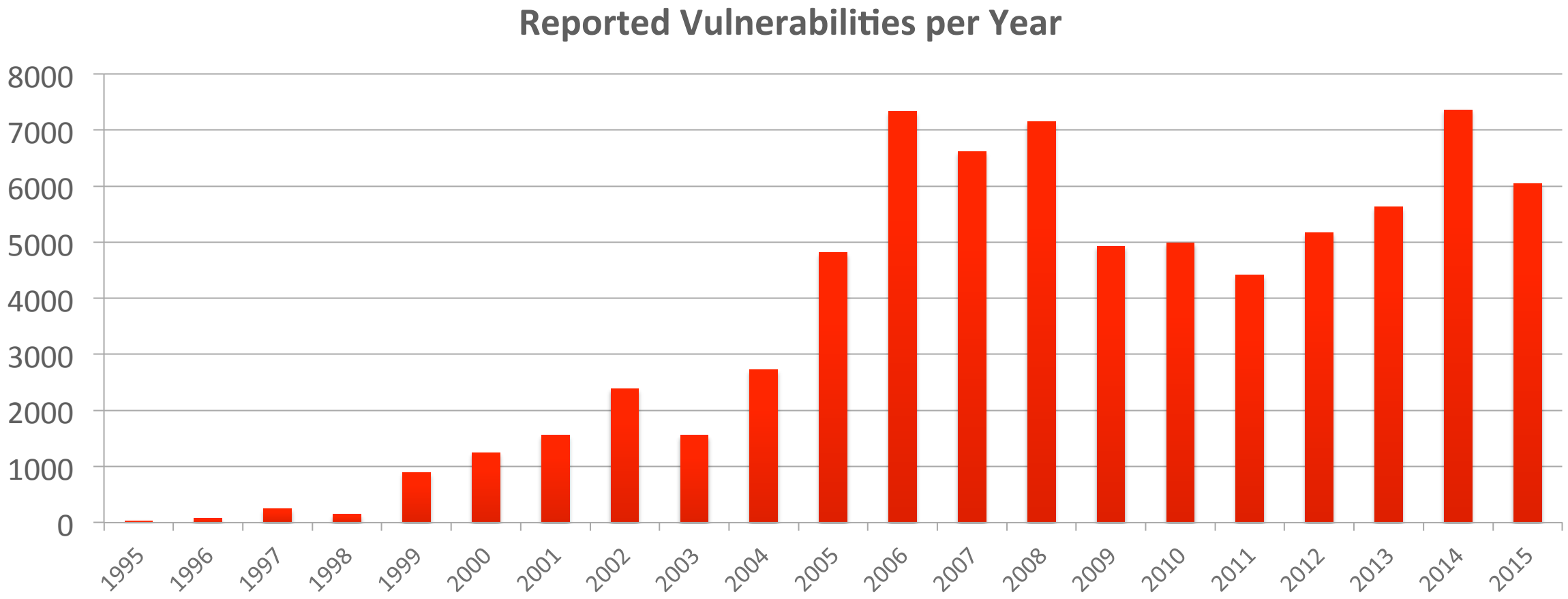
[1] Computer Security Technology Planning Study, 1972. [2] Phrack Magazine, 8(54), article 8

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

How is this possible?

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

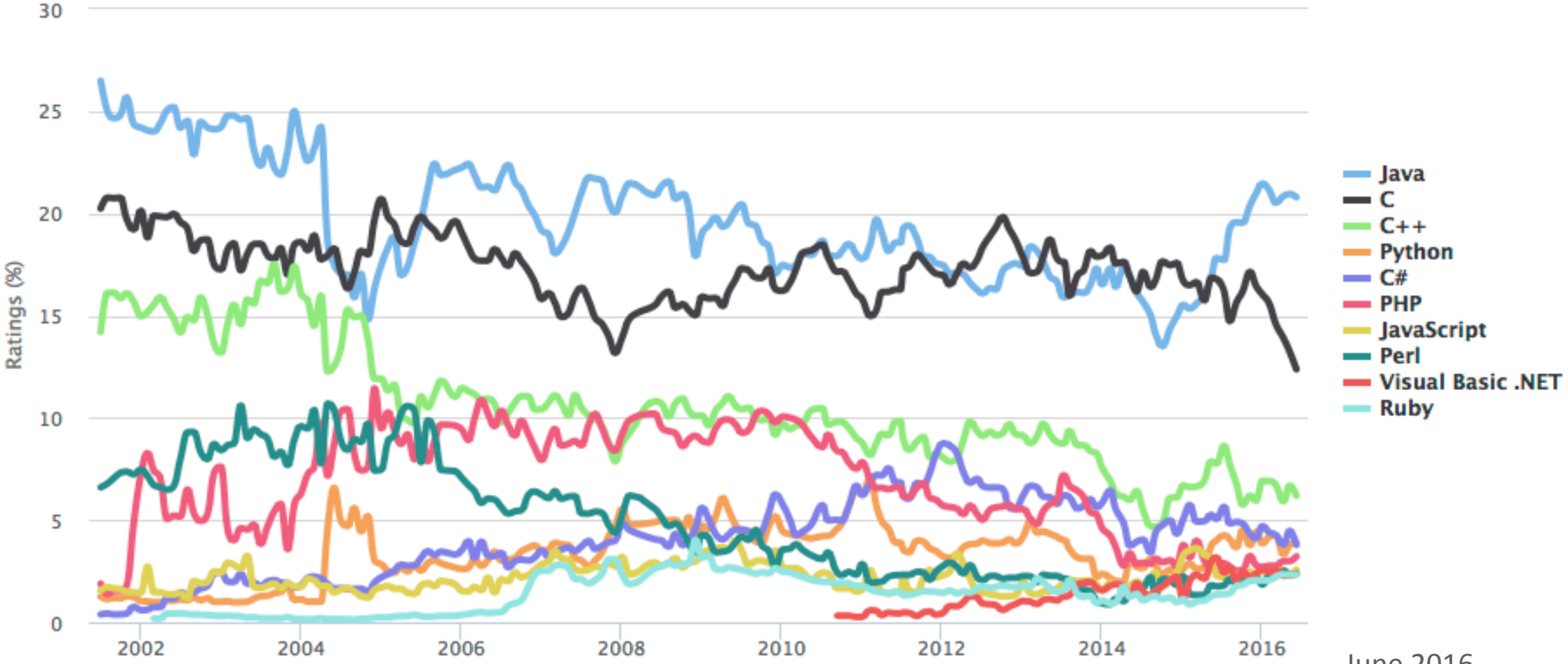


National Vulnerability Database, http://nvd.nist.gov

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

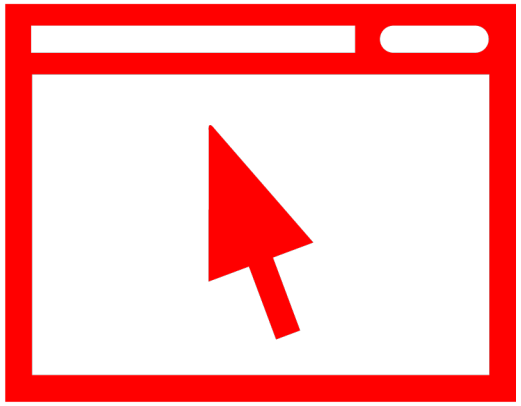
TIOBE Programming Community Index

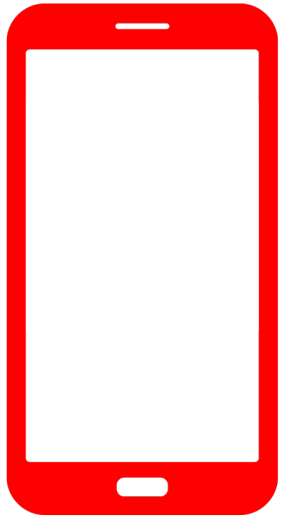
Source: www.tiobe.com

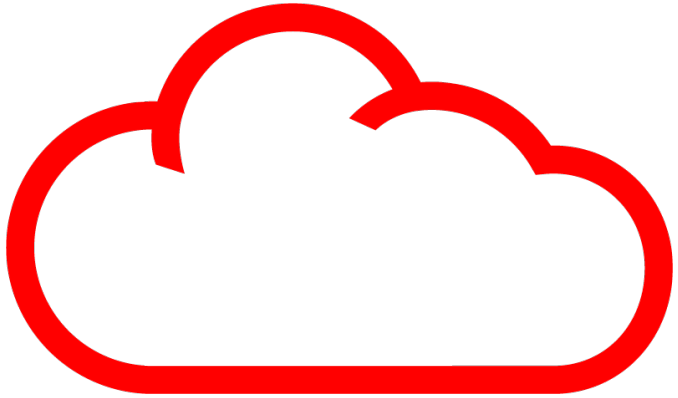


June 2016

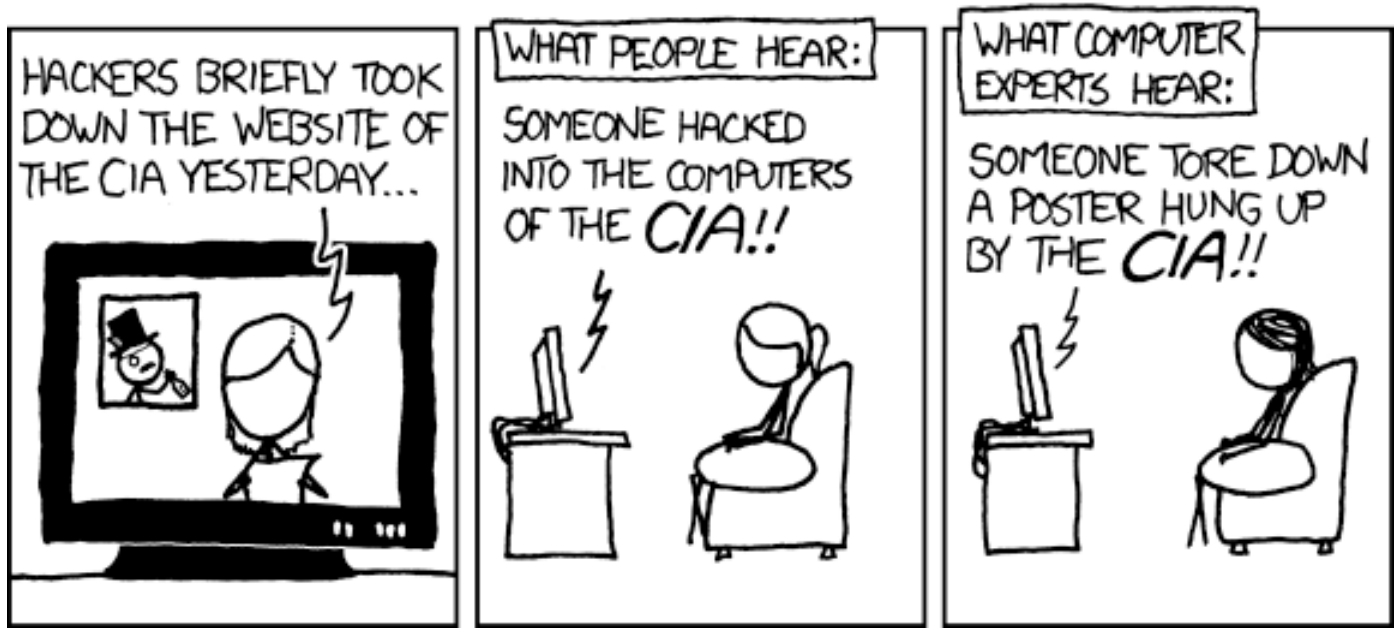
Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |





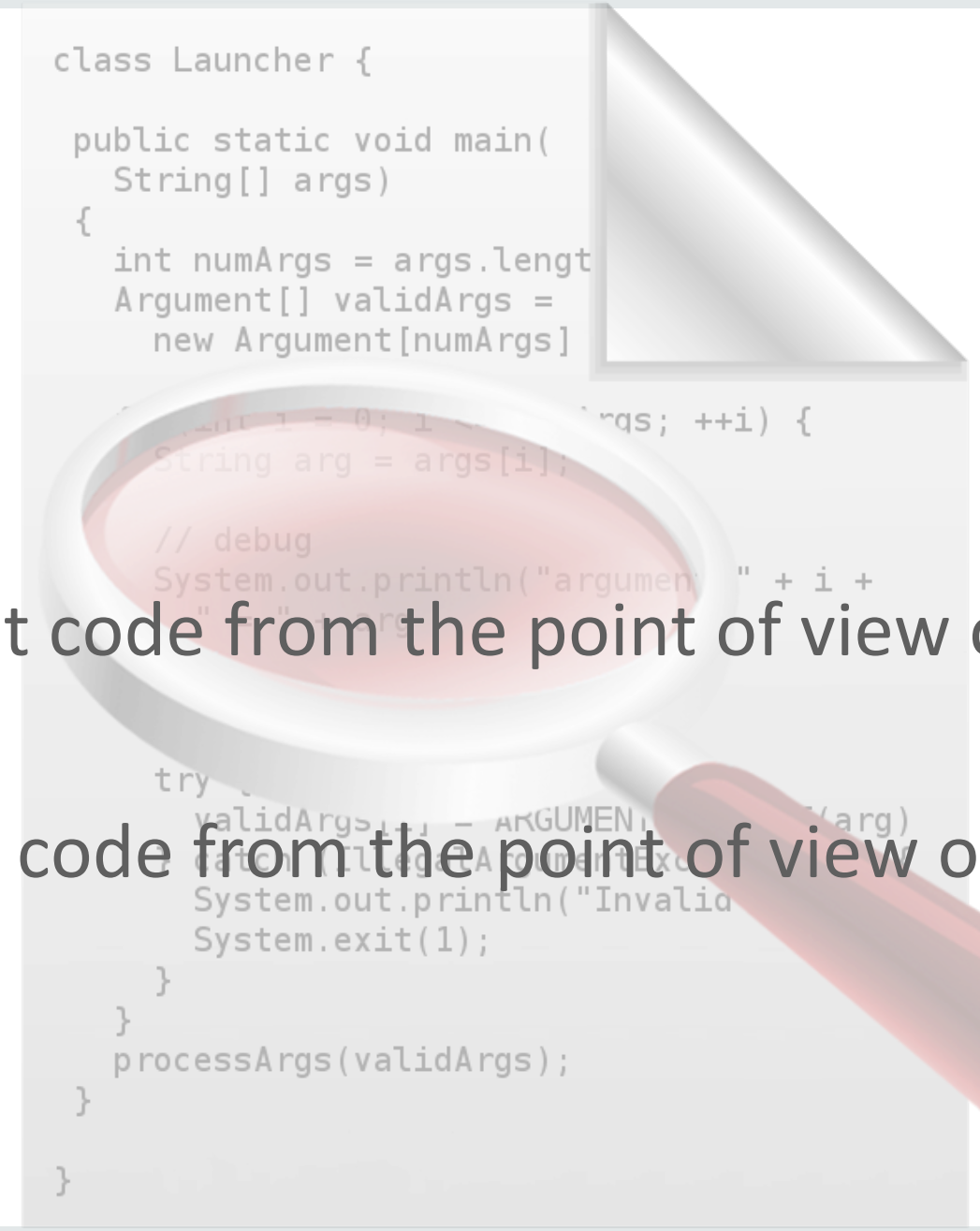


Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



http://xkcd.com/932/

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



Attackers look at code from the point of view of how to **break** into it

Developers write code from the point of view of **functionality** required

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Sample Attacker Techniques

- **Buffer Errors**
  - Return to stack
  - RoP
  - Heap spray
  - Return to libc
  - …

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Sample Attacker Techniques

- **Buffer Errors**
  - Return to stack
  - RoP
  - Heap spray
  - Return to libc
  - …
- **Injections**
  - Missing validation altogether or validating SQL as HTML
  - Edge cases (e.g., partial sanitisation of HTML entities)
  - Use of blacklists rather than whitelists
  - …

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Sample Attacker Techniques

- **Buffer Errors**
  - Return to stack
  - RoP
  - Heap spray
  - Return to libc
  - …
- **Injections**
  - Missing validation altogether or validating SQL as HTML
  - Edge cases (e.g., partial sanitisation of HTML entities)
  - Use of blacklists rather than whitelists
  - …
- **Access Control**
  - Missing checks
  - Incorrect checks (e.g., check for “logged in user” and not “logged in administrator”)

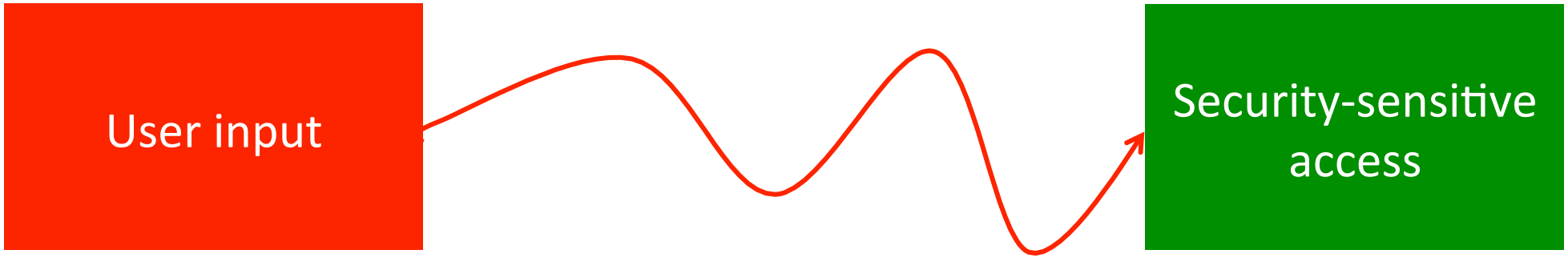
ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Why Is This Happening?

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Buffer Errors
Injections
Access Control



data or code

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Need Sanitisation

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Sanitisation is the developer's responsibility

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Most languages do not provide sanitisation support

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Solution: Education

- Many PL subjects don’t cover security aspects per se
- Few universities offer PL Security courses at undergraduate level
- Companies do own training

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Solution: Education

- Many PL subjects don't cover security aspects per se
- Few universities offer PL Security courses at undergraduate level

Not Working

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Solution: Static Analysis Tools

- Re-parse the code with different compiler
- Encode semantics of PL in intermediate representation (IR)
- Analyse IR with over or under approximations
- Report bugs/vulnerabilities with False Positives (over) or False Negatives (under)

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Solution: Static Analysis Tools

- Re-parse the code with different compiler
- Encode semantics of PL in intermediate representation (IR)
- Analyse IR with over or under approximations
- Report bugs/vulnerabilities with False Positives (over) or False Negatives (under)

Incomplete

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Solutions: Dynamic and Verification Tools

Dynamic

- Dynamic analysis tools
  - Instrumentation
  - Introspection
- Fuzzing

Verification

- Model checking
- Theorem proving

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Solutions

Dynamic

- Dynamic analysis tools
  - Instrumentation
  - Introspection
- Fuzzing

Incomplete

Verification

- Model checking
- Theorem proving

Doesn't scale

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

We can design languages that avoid some of these issues statically

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Avoid Buffer Errors Statically

Graydon Hoare, 2009

- Guaranteed memory safety
  - Ownership
  - Borrowing
    shared borrow (`&T`)
    mutable borrow (`&mut T`)
- Efficiency
  - Zero-cost abstractions
  - Parallelisation

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

We can design languages or extensions that track some of these issues dynamically

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Avoid Buffer Errors Statically and Dynamically

- Extends C with bounds checking
  - 3 new checked pointer types: `ptr`, `array_ptr`, `span`
  - Bounds-checked arrays: `checked`
  - Checked member bounds
  - Bounds-safe interfaces
  - Checked program scopes
  - Lightweight invariants
  - Runtime errors on pointer arithmetic overflow and null pointer for `array_ptr`
- Dynamic checks

David Tarditi, June 2016 (v 0.5)

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Avoid Buffer Errors Dynamically

John McCarthy, 1958

- Managed memory
  - Garbage collection
  - First introduced in LISP in 1958
- Now in
  - OO languages: Smalltalk, Java, C#, JavaScript, Go
  - Functional languages: ML, Haskell, APL
  - Dynamic languages: Ruby, Perl, PHP

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Avoid Injections Dynamically

Larry Wall, 1987

- Taint mode
  - Perl 3, 1989
  - Automatic checks when program running with different real and effective user or group IDs
  - -T flag to turn it on
- Similar ideas in
  - Ruby

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

We have made first steps to provide developers with access control support in the language

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

First Steps at Avoiding Access Control Issues

James Gosling, 1991

- Safely run untrusted applets on your computer
- Security model
  - Java 1.2, 1998 [Li Gong]
  - Subscribes to the principle of least privilege
  - Security Manager mediates all access control decisions
  - Stack-based checks
- Similar ideas in
  - .NET framework

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

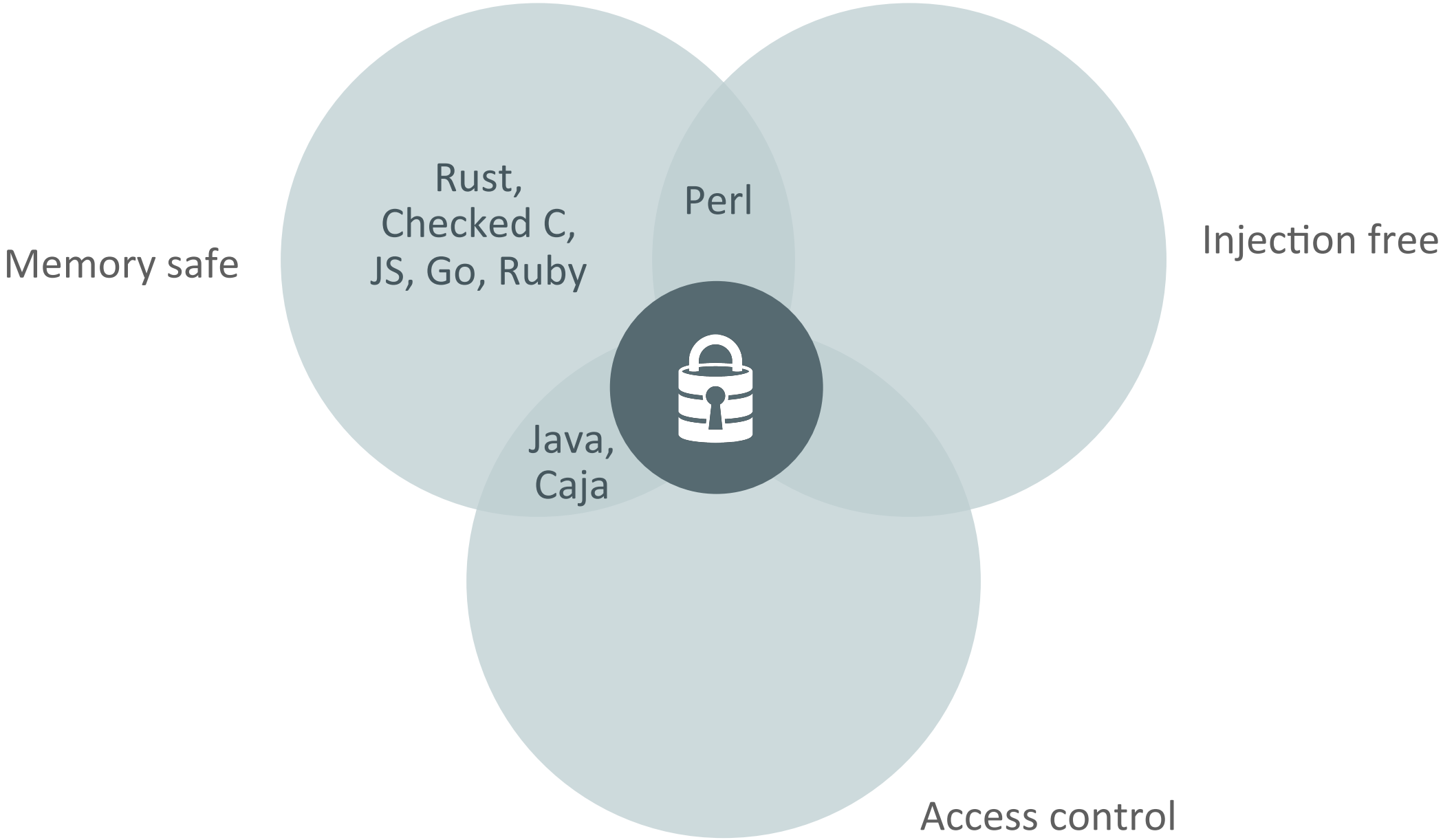
Avoid Access Control Issues

Mark Miller, ~2010

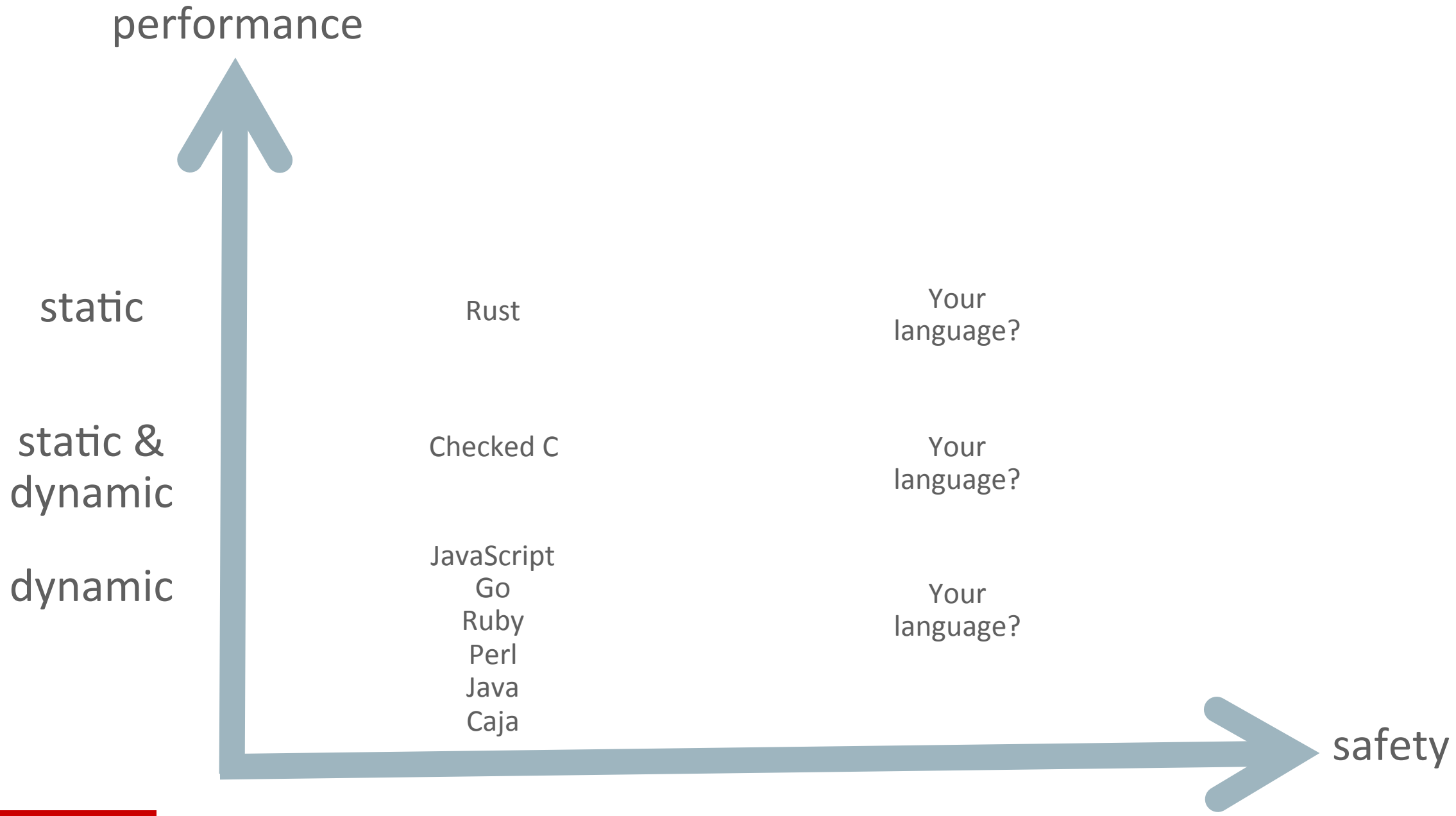
- Extends/limits JavaScript to safely embed untrusted active content inside your web browser
  - Available as a plugin, ~2011
- Object-oriented capabilities
  - A capability-secure JS subset (SES – Secure ECMAScript)
  - A safe DOM wrapper (Domado)
  - A HTML and CSS sanitiser

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



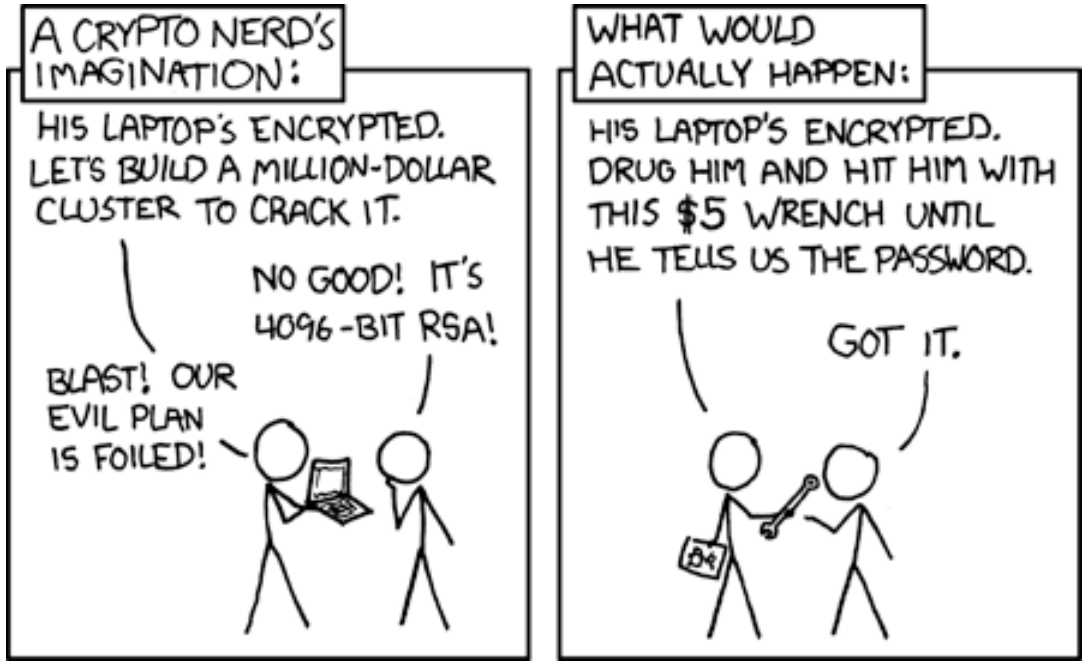
Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

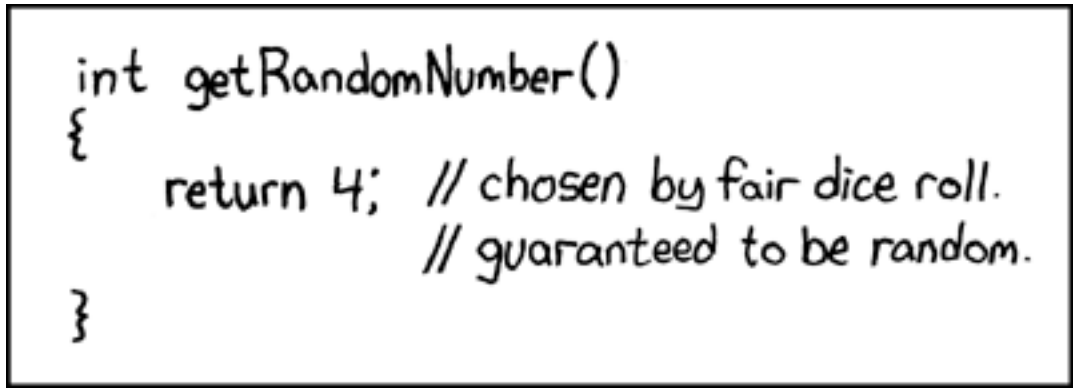
Intermezzo

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



http://xkcd.com/538/

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



http://xkcd.com/221/

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

1769

Vulnerabilities due to **cryptographic issues** (2013-2015)

National Vulnerability Database, http://nvd.nist.gov

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Cryptographic Issues

- Use of hardcoded passwords
- Use of deprecated algorithms
- Use of wrong defaults
- Use of hardcoded seeds
- Improperly hashed passwords
- Deterministic seeds to generate random numbers
- Valid users and host security keys left on an image of a cloud platform
- …

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Sample Attacker Techniques

- Poor API design (e.g., weak defaults)
- Weaknesses in protocols
- Use of deprecated suites
- “Breakthroughs”

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

We have made first steps at providing a Cryptographic API for non-crypto developers to use

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

First Steps to Avoid Cryptographic Issues

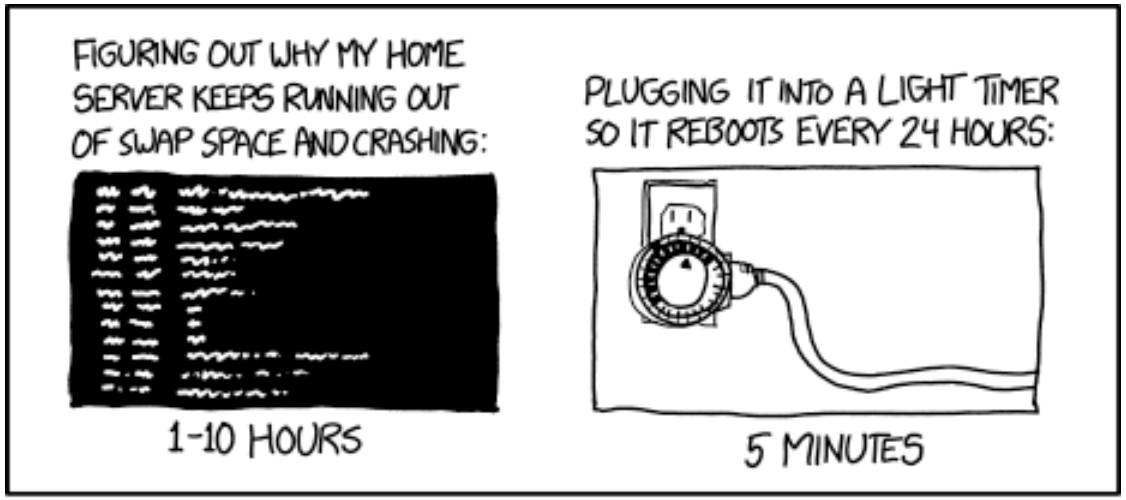
James Gosling, 1991

- Cryptography Architecture
  - Java 1.1, 1997
  - Framework for cryptography
  - APIs for encryption, key generation & management, secure random number generation, certificate validation, …
- Similar ideas in
  - .NET framework
  - PyCrypto
  - krypt for Ruby
  - crypto for Go, …

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Crypto APIs are too low level

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

894

Vulnerabilities due to **resource management** issues (2013-2015)

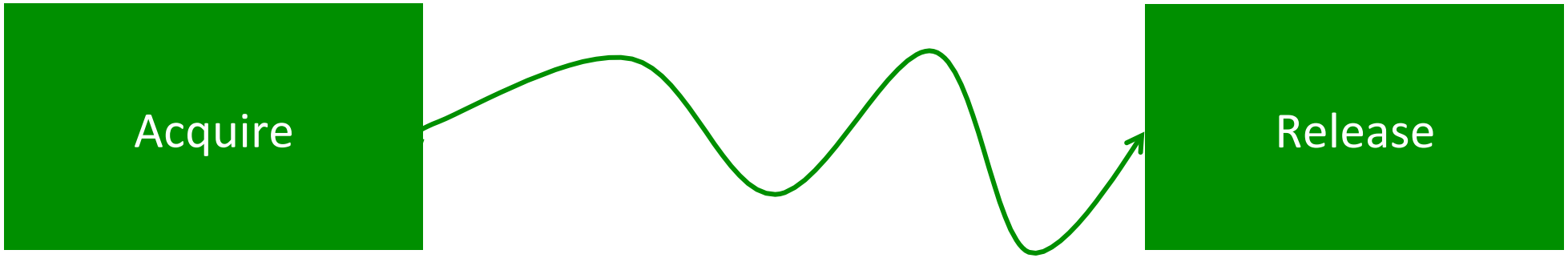
Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Resource Management Issues

- Use after free
- Double free
- Memory corruption
- Type casting error
- Worker termination error

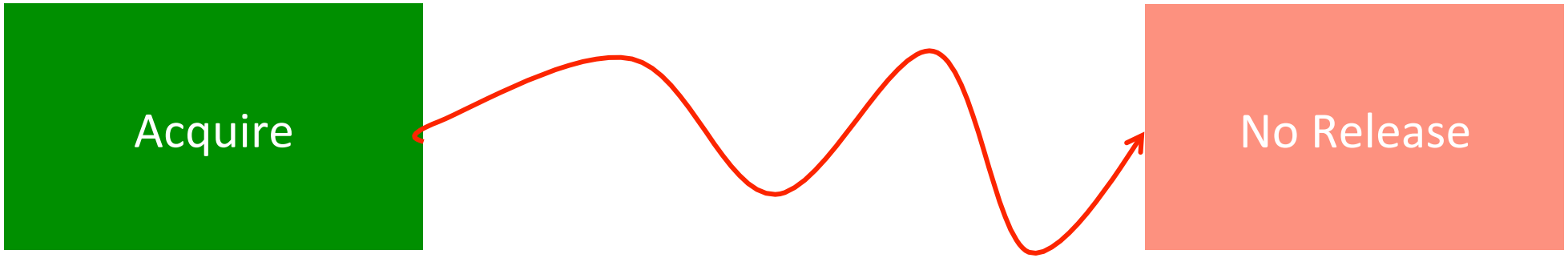
Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Resource Management Issues



Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Resource Management Issues



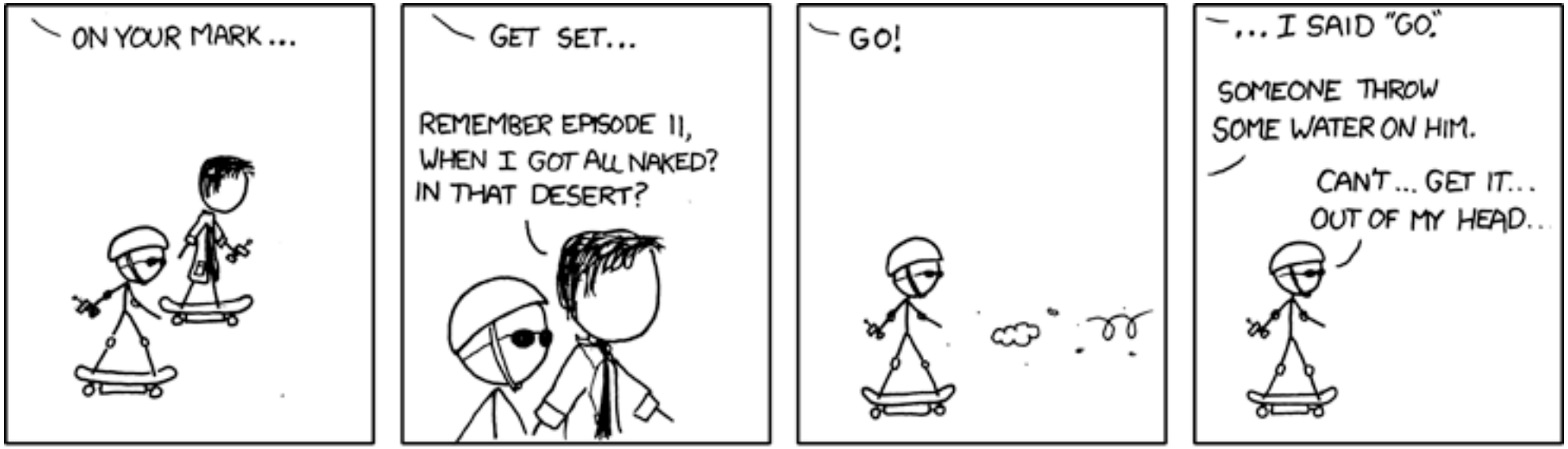
Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Few solutions to resource management issues are available

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Memory Safe languages avoid use-after-free and double-free issues

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



http://xkcd.com/580/

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

149

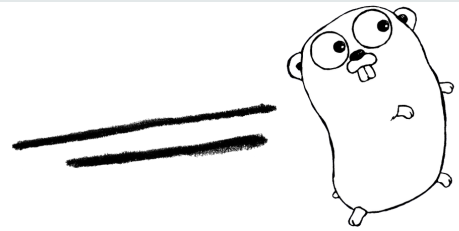
Vulnerabilities due to **race conditions** (2013-2015)

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

We need to consider the future concurrent world and make race conditions a thing of the past

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Concurrency by Default

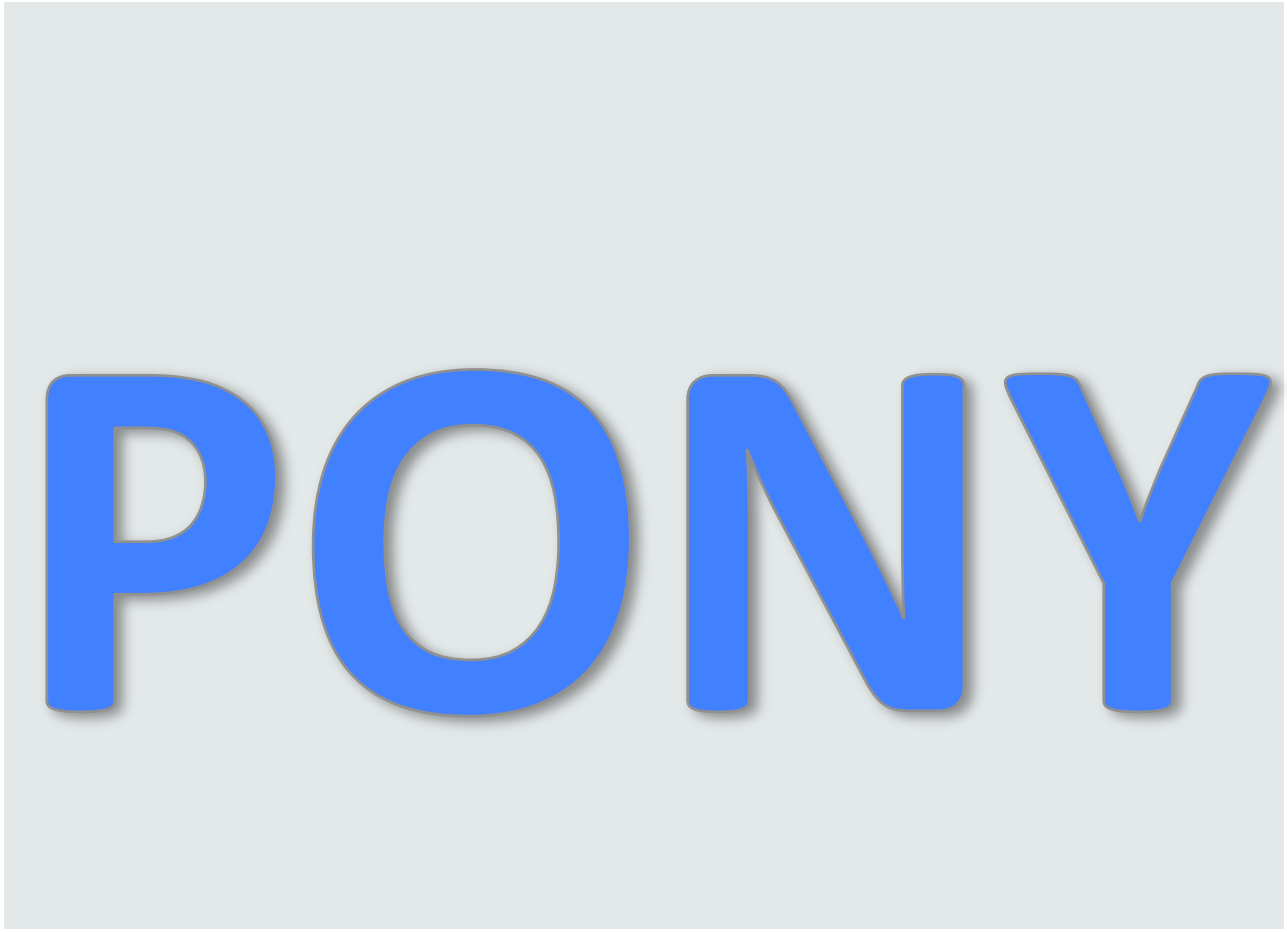


- Goroutines for concurrency
  - CSP-style, light-weight process
  - `go`routines communicate and synchronise using `chan`nels
- Main goals
  - simplicity, safety and readability
- Not data-race free

Robert Griesemer, Rob Pike and Ken Thompson, 2007

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Avoid Race Conditions



Sylvan Clebsch, Sebastian Blessing, Sophia Drossopoulou, ~2014

- Actors for concurrency
  - Fast: zero-copy messaging
  - Safe: **data-race free type system**
- Reference capabilities/Type qualifiers based on deny properties
  - iso, trn, ref, val, box, tag

  attached to the path to an object
  - mutable, immutable, opaque

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

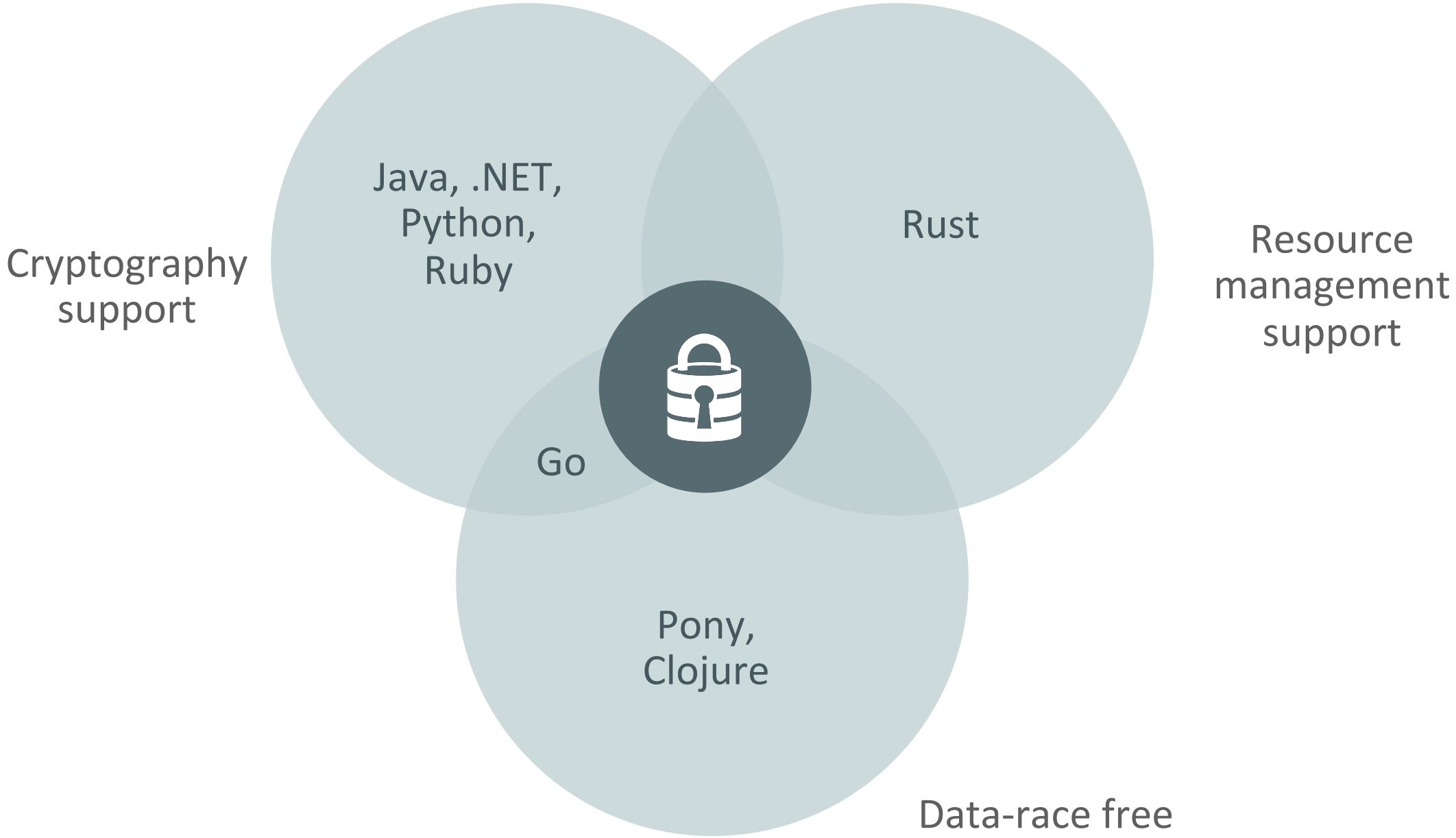
Avoid Race Conditions

- Concurrency via immutable data structures
  - Identities: a series of immutable states over time

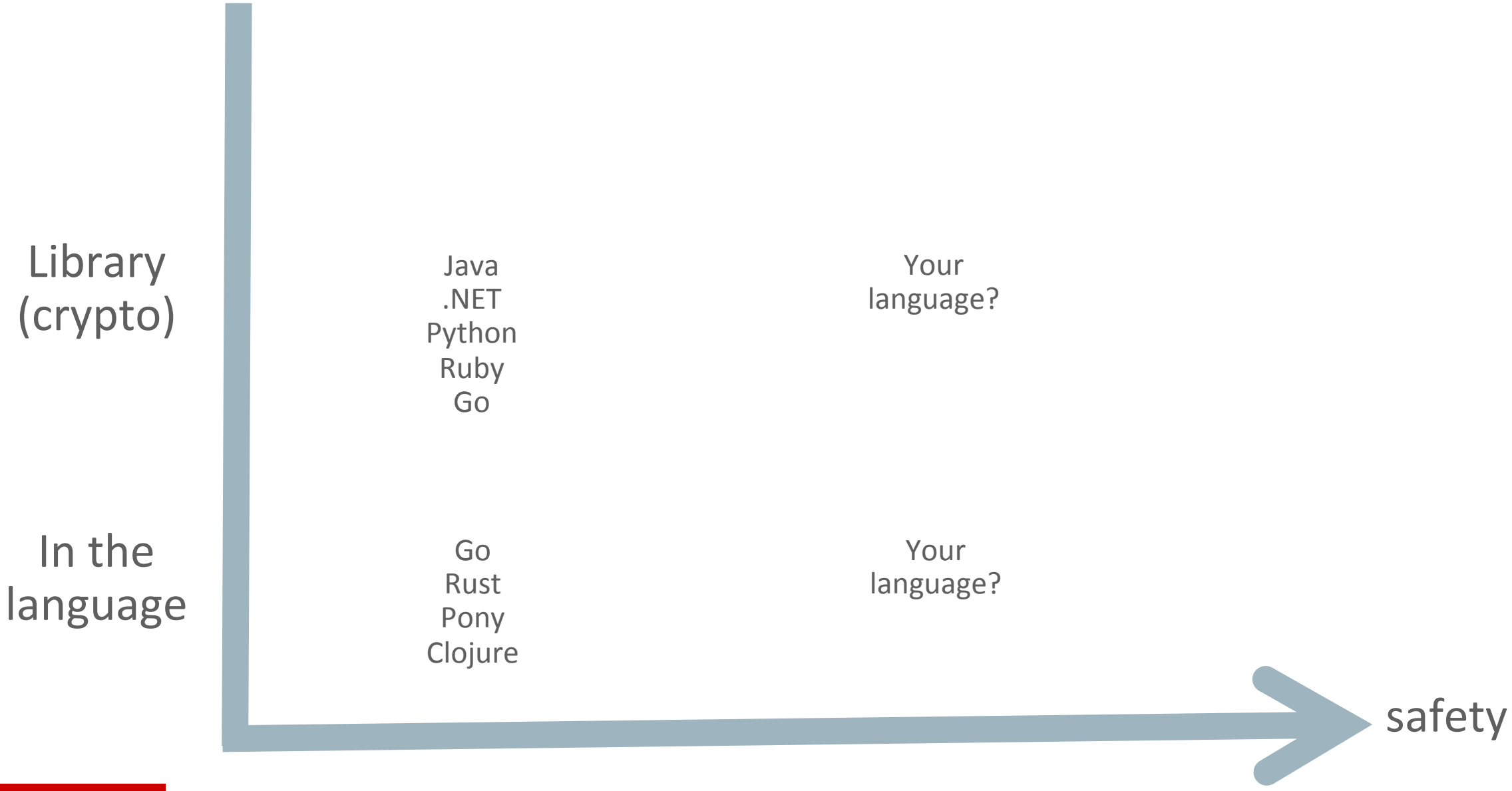
  and mutable reference types
- Designed for simplicity and data orientation

Rick Hickey, ~2005

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



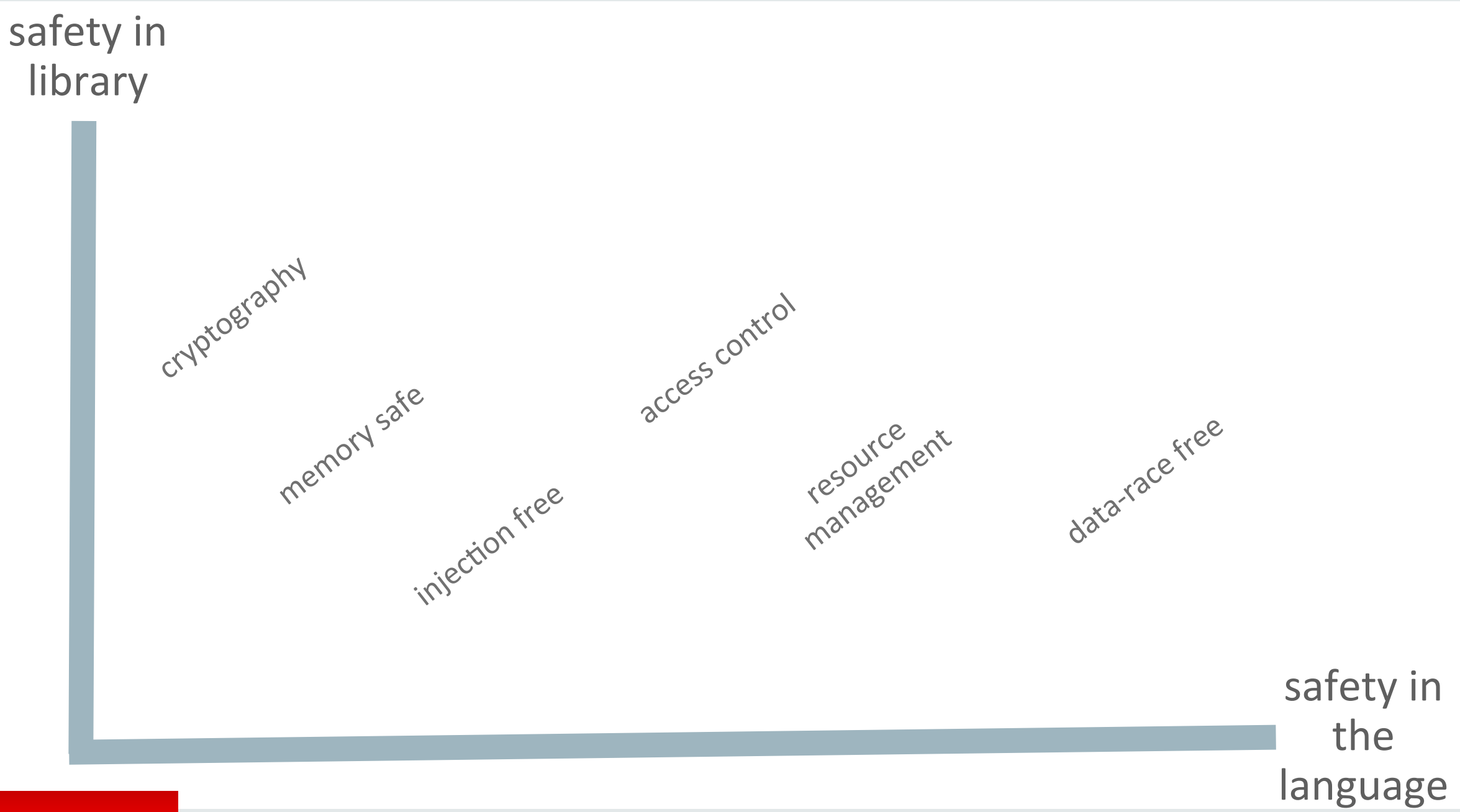
Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



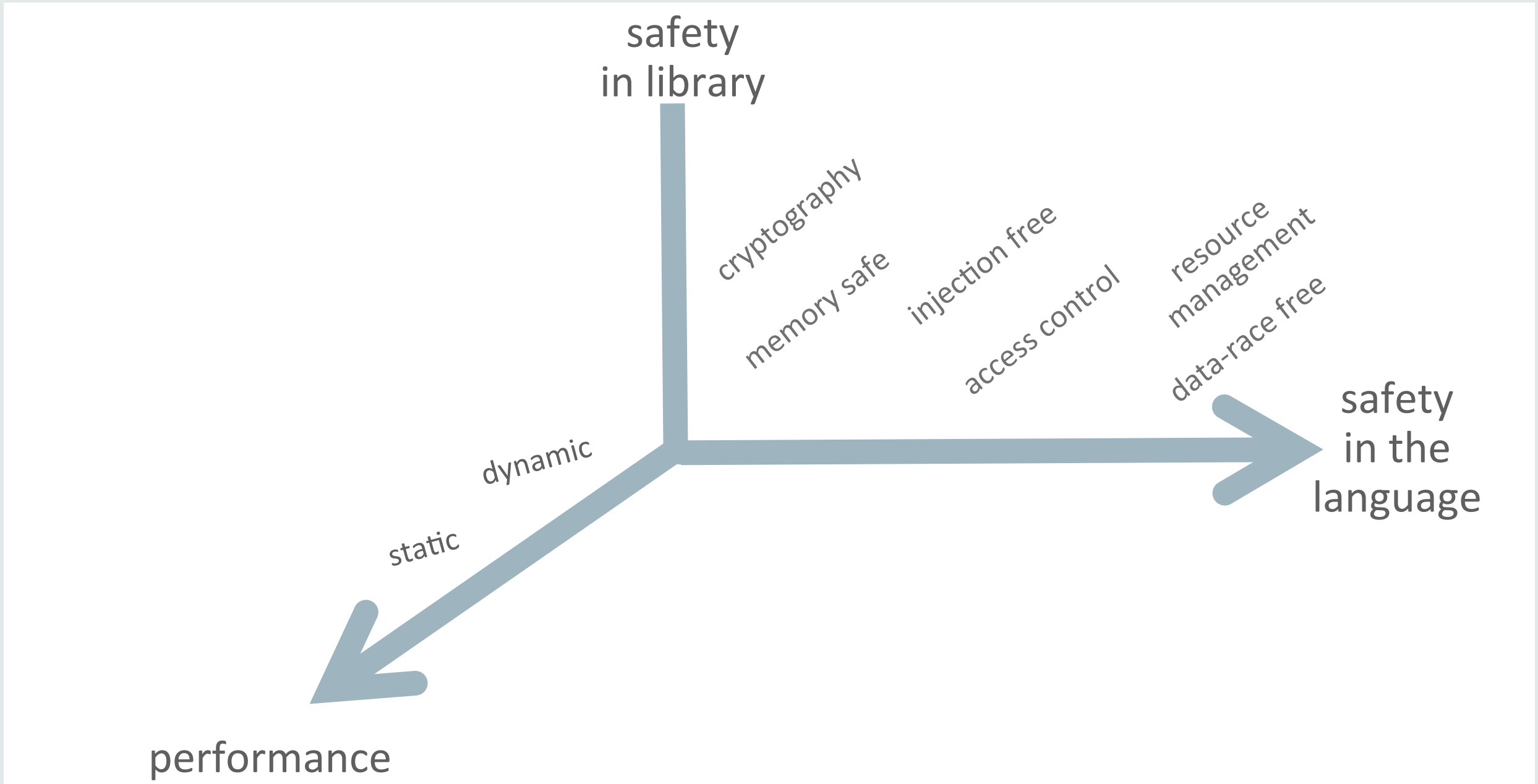
Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

What is Security in the Context of Programming Languages?

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

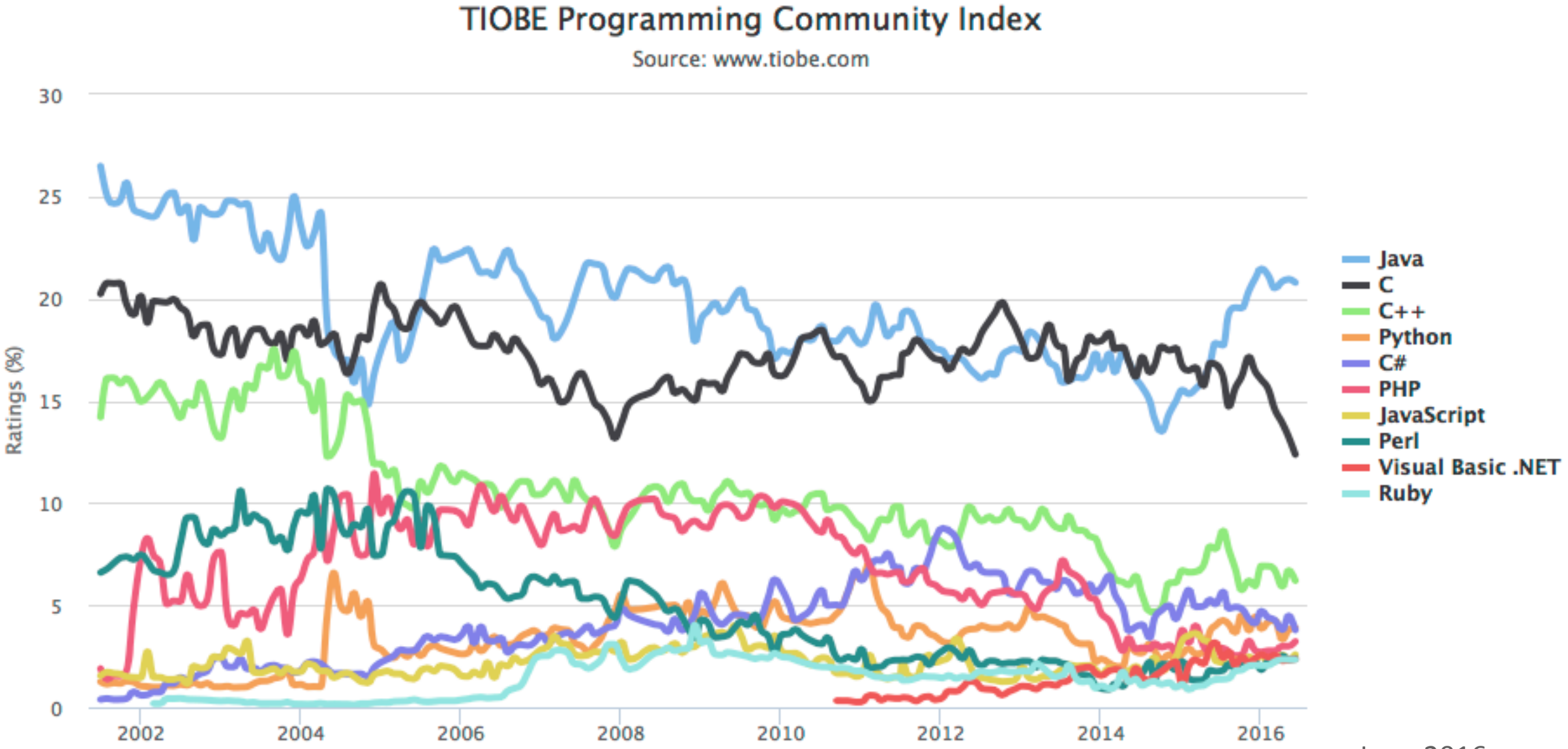


Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

14844

Vulnerabilities in the past 3 years (2013-2015)

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |



June 2016

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Education lagging

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

It’s time to include security in our language design

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

There Will be Barriers for New Adoption

- Performance
- Usability
- Removing cognitive overload
- Legacy language support / interoperability / FFI

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Challenge – To design languages that provide security and eradicate buffer errors, injections, access control issues, cryptography issues, resource management issues and race conditions

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Challenge – To provide the context for tainted data that crosses along different layers

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Challenge – To provide high-level crypto APIs (e.g., stores password, does hashing) that don't require changes over time?

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Challenge – To provide security guarantees in the languages we design

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

18.5

million software developers worldwide (11M professional, 7.5M hobbyist)

http://www.idc.com/research/viewtoc.jsp?containerId=244709, IDC December 2013 report

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Security in not just for expert developers

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

We need security for the masses

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

cristina.cifuentes@oracle.com

http://labs.oracle.com/locations/australia

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Integrated Cloud

Applications & Platform Services

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |


ORACLE®