
ORACLE®

It’s Time For Secure Languages

Cristina Cifuentes
Oracle Labs Australia
October 2017

The following is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.

25 exploited vulnerabilities in 1995

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

27287

exploited vulnerabilities in 2013-2016

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

3781

exploited vulnerabilities due to **buffer errors** (2013-2016)

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

2861

exploited vulnerabilities due to **cross-site scripting** (2013-2016)

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

2094

exploited vulnerabilities due to **information leak** (2013-2016)

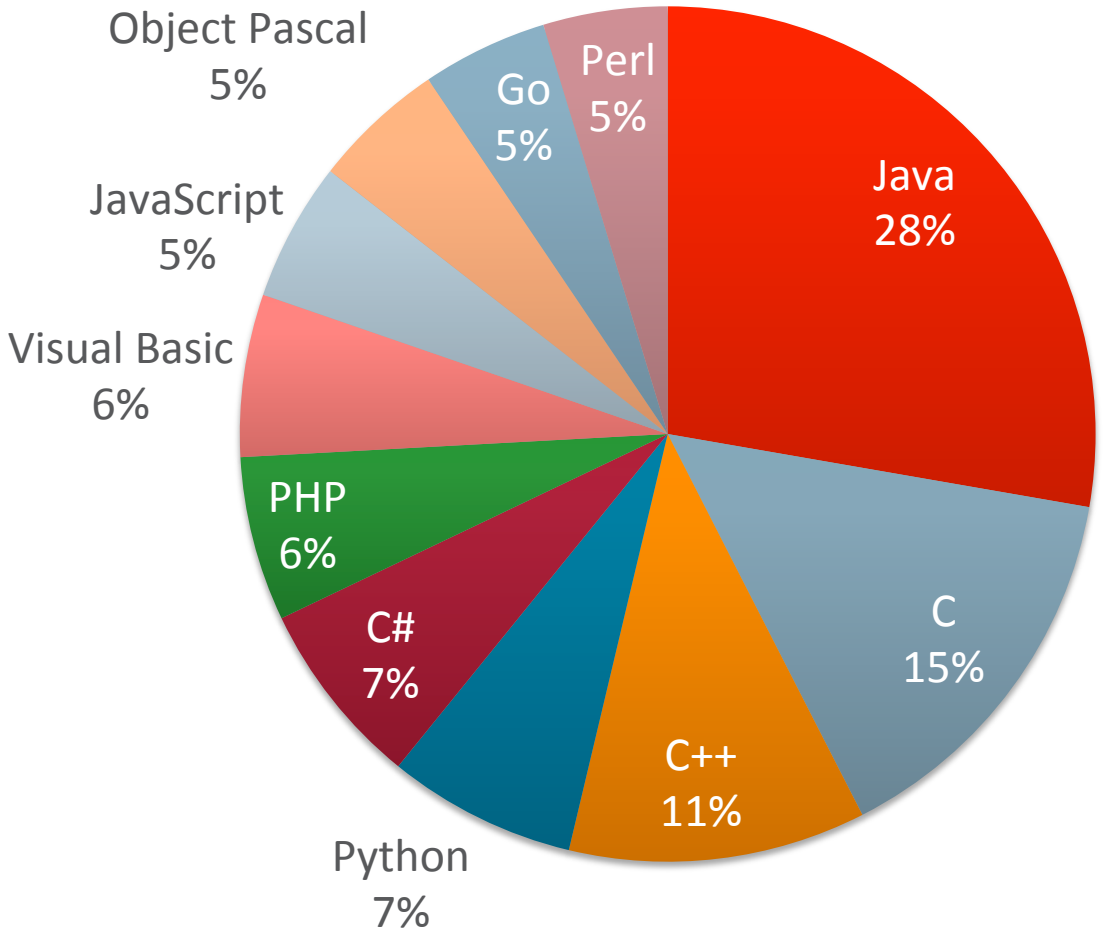
National Vulnerability Database, http://nvd.nist.gov

50%

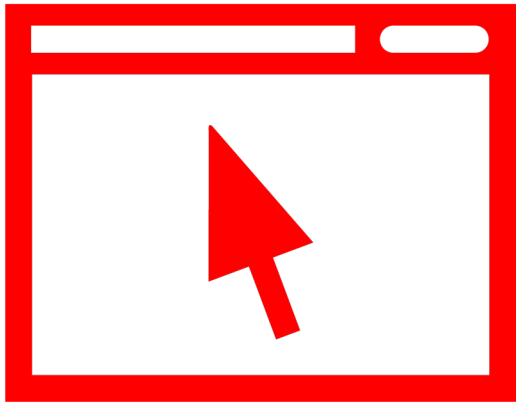
exploited vulnerabilities in NVD were buffer errors, injections and information leak (2013-2016)

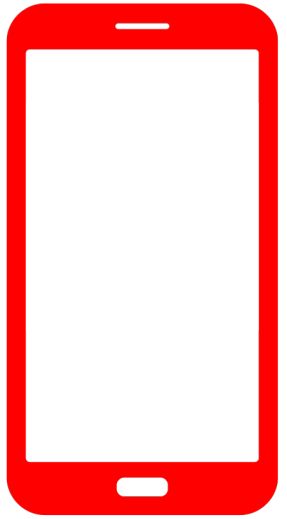
National Vulnerability Database, http://nvd.nist.gov

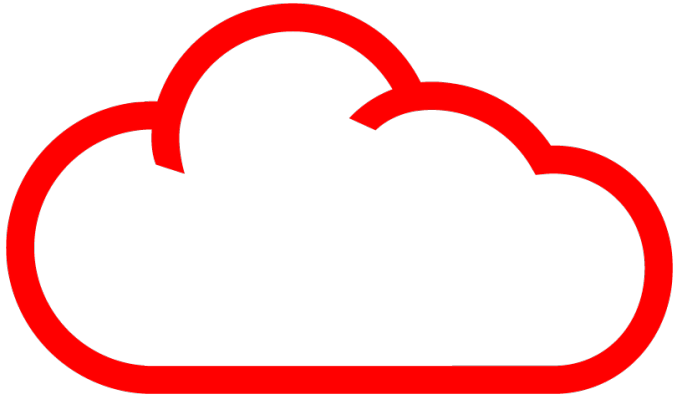
Top Mainstream Languages



Tiobe index, July 2017





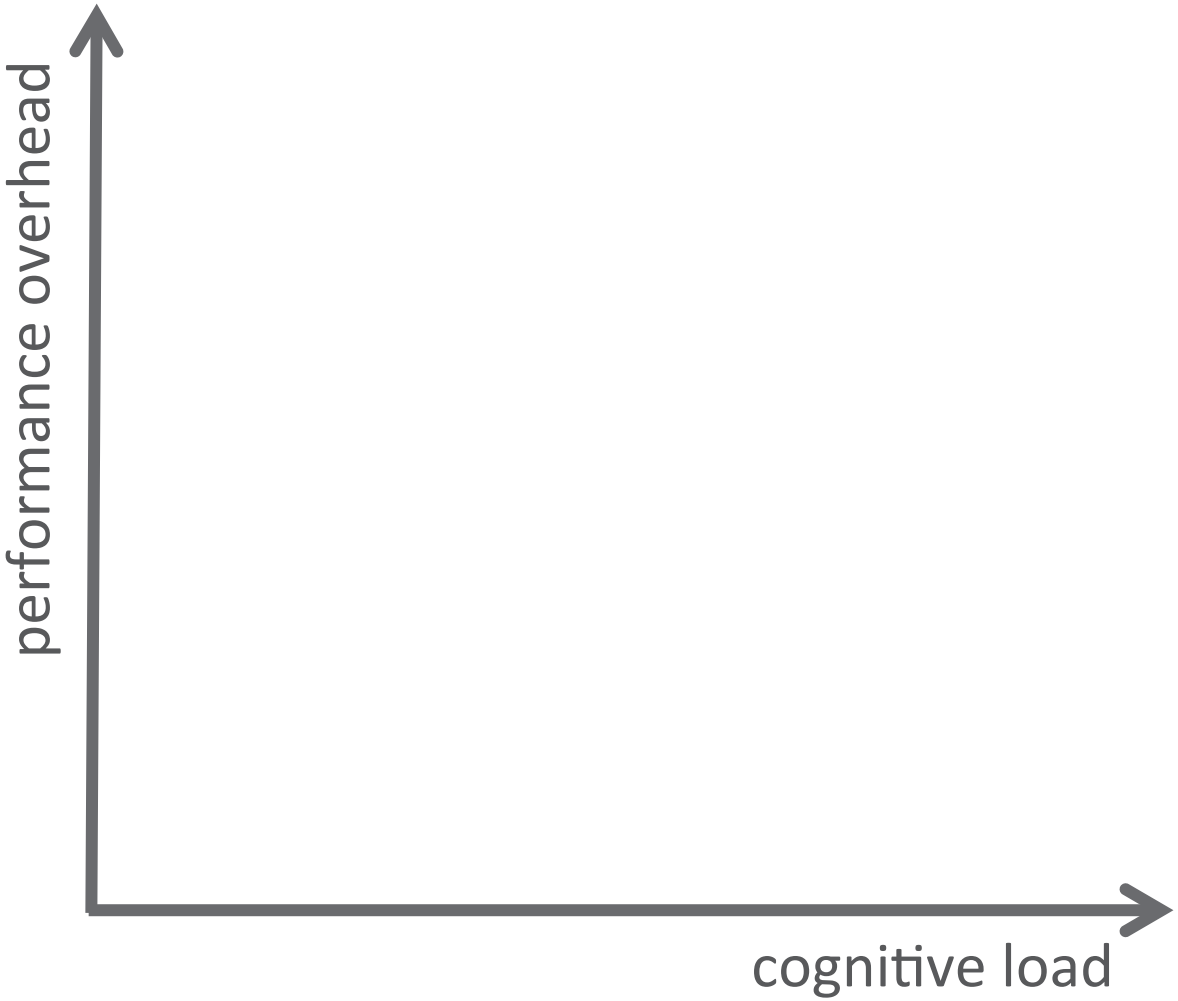


$4M

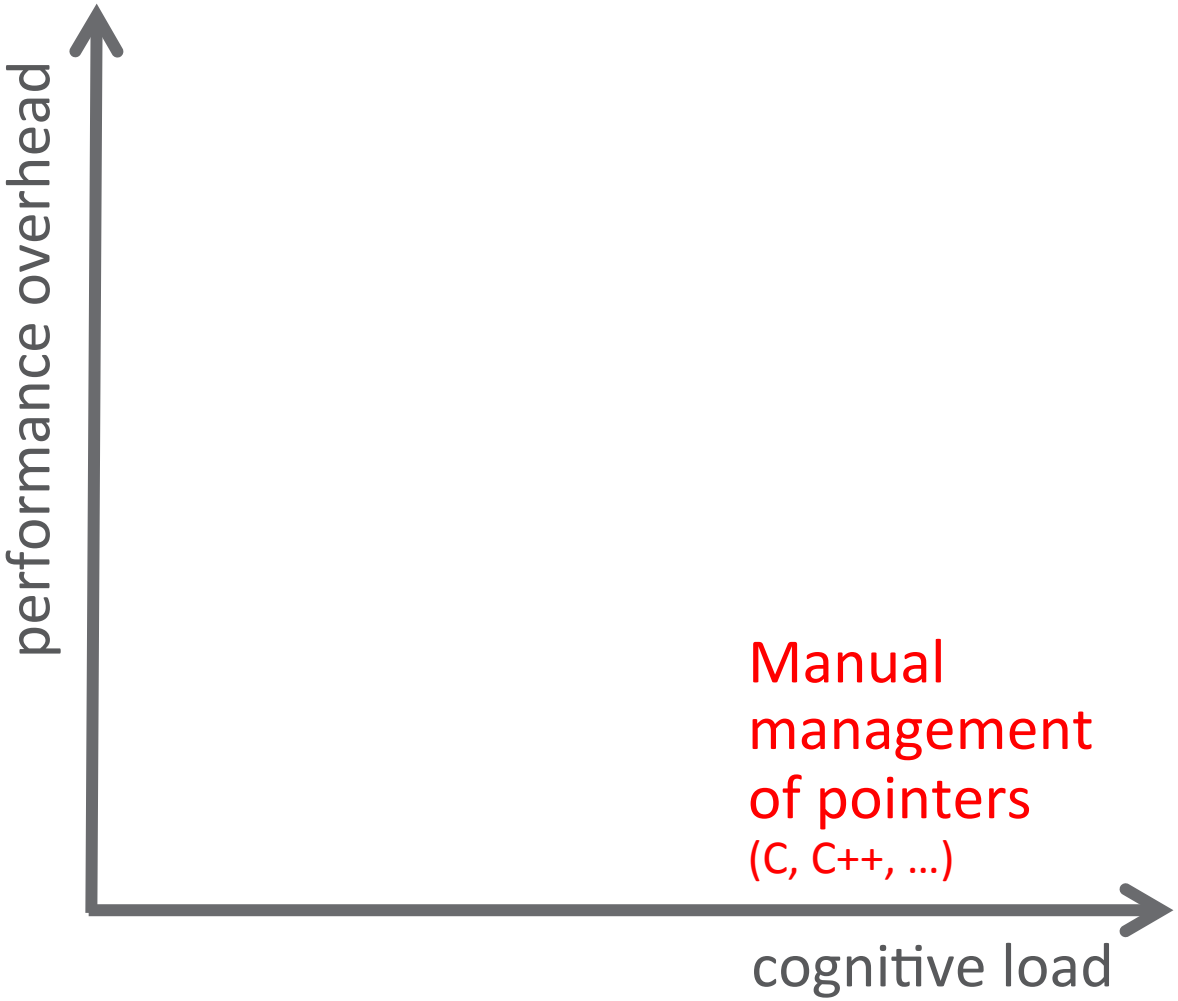
average cost of a data breach

2016 Ponemon Cost of Data Breach Study

Why Is This Happening?



Buffer Errors – The Problem: Unsafe Abstraction



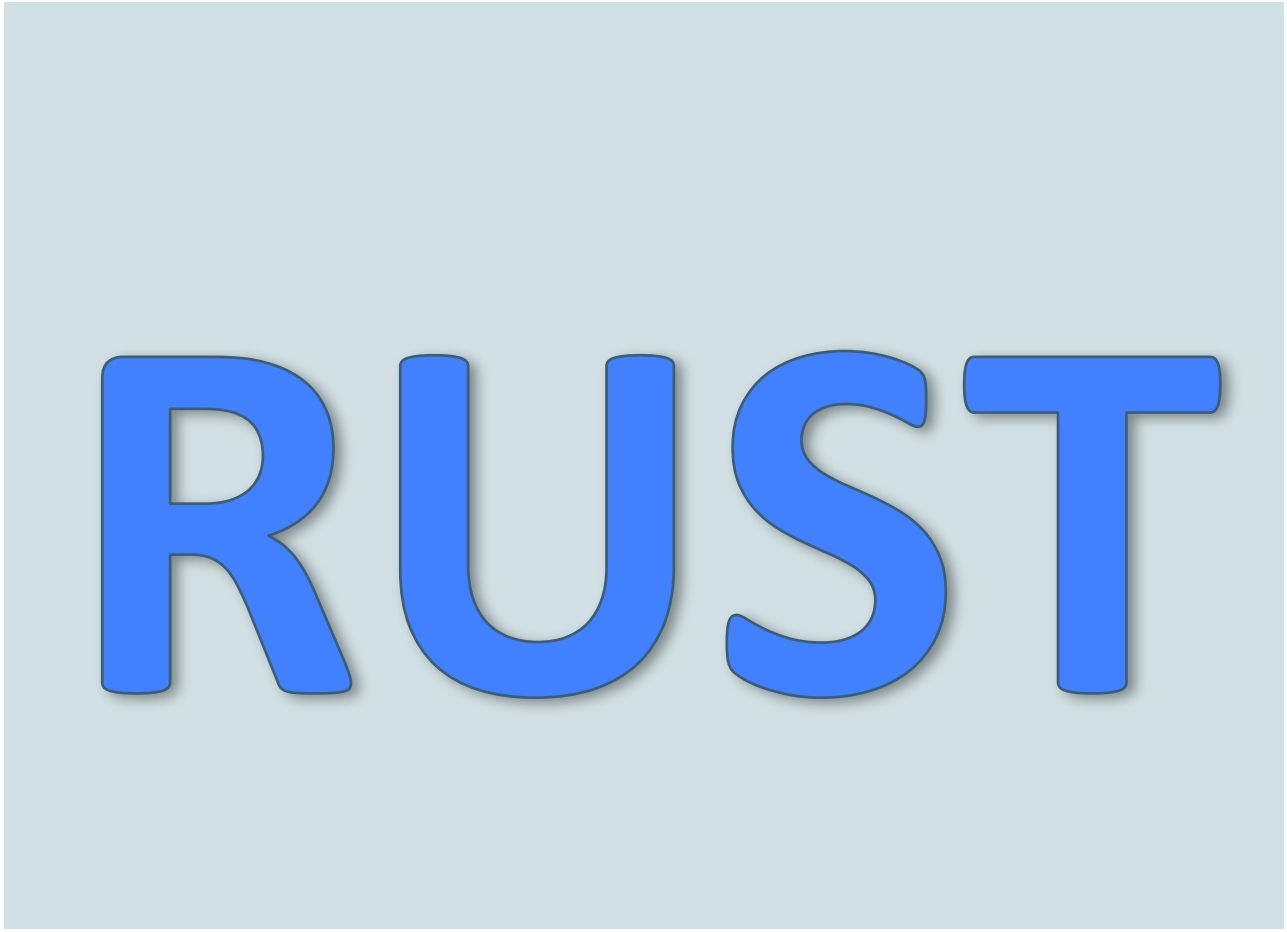
We have designed languages that prevent buffer errors

Avoid Buffer Errors Dynamically

John McCarthy, 1958

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

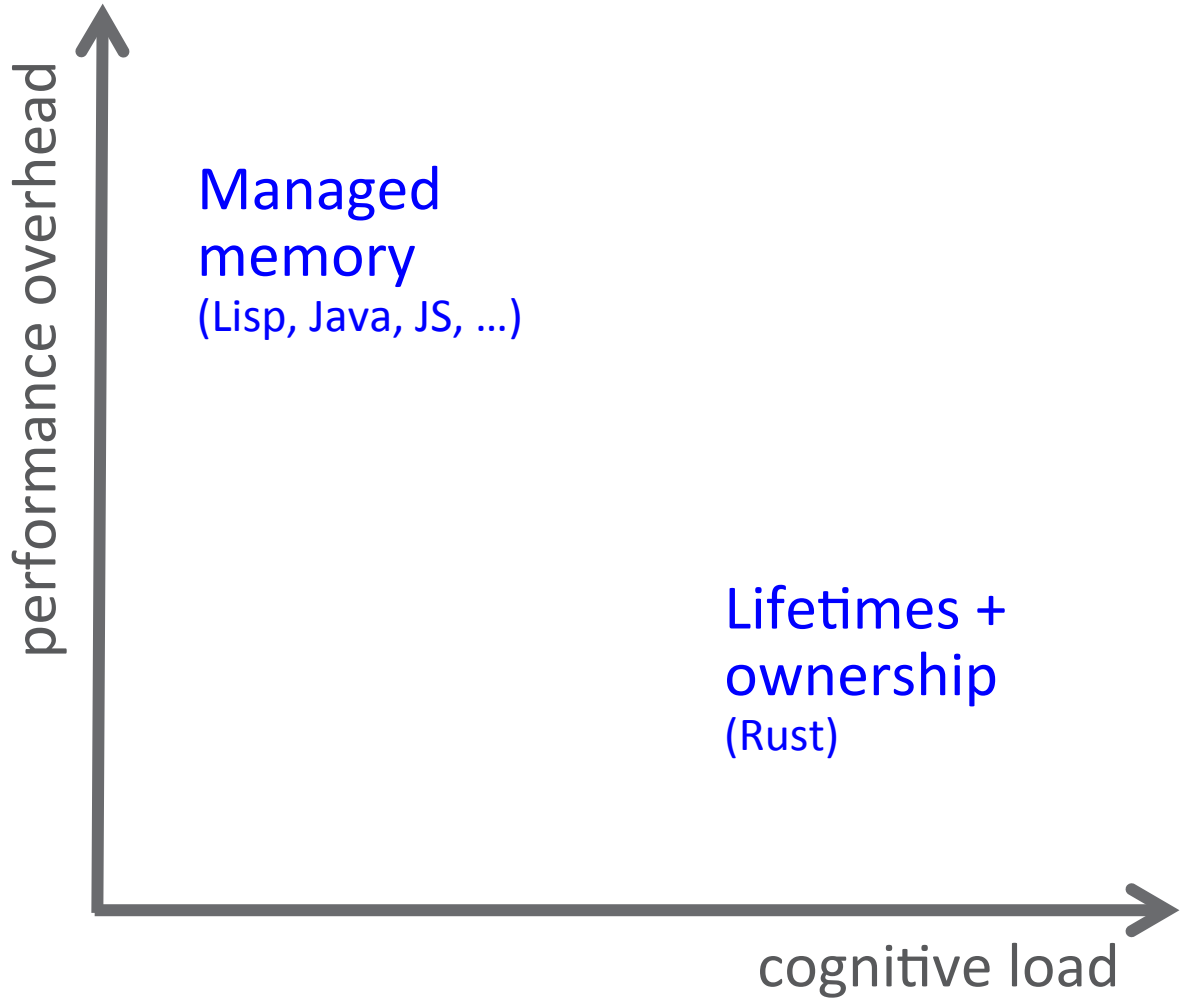
Avoid Buffer Errors Statically



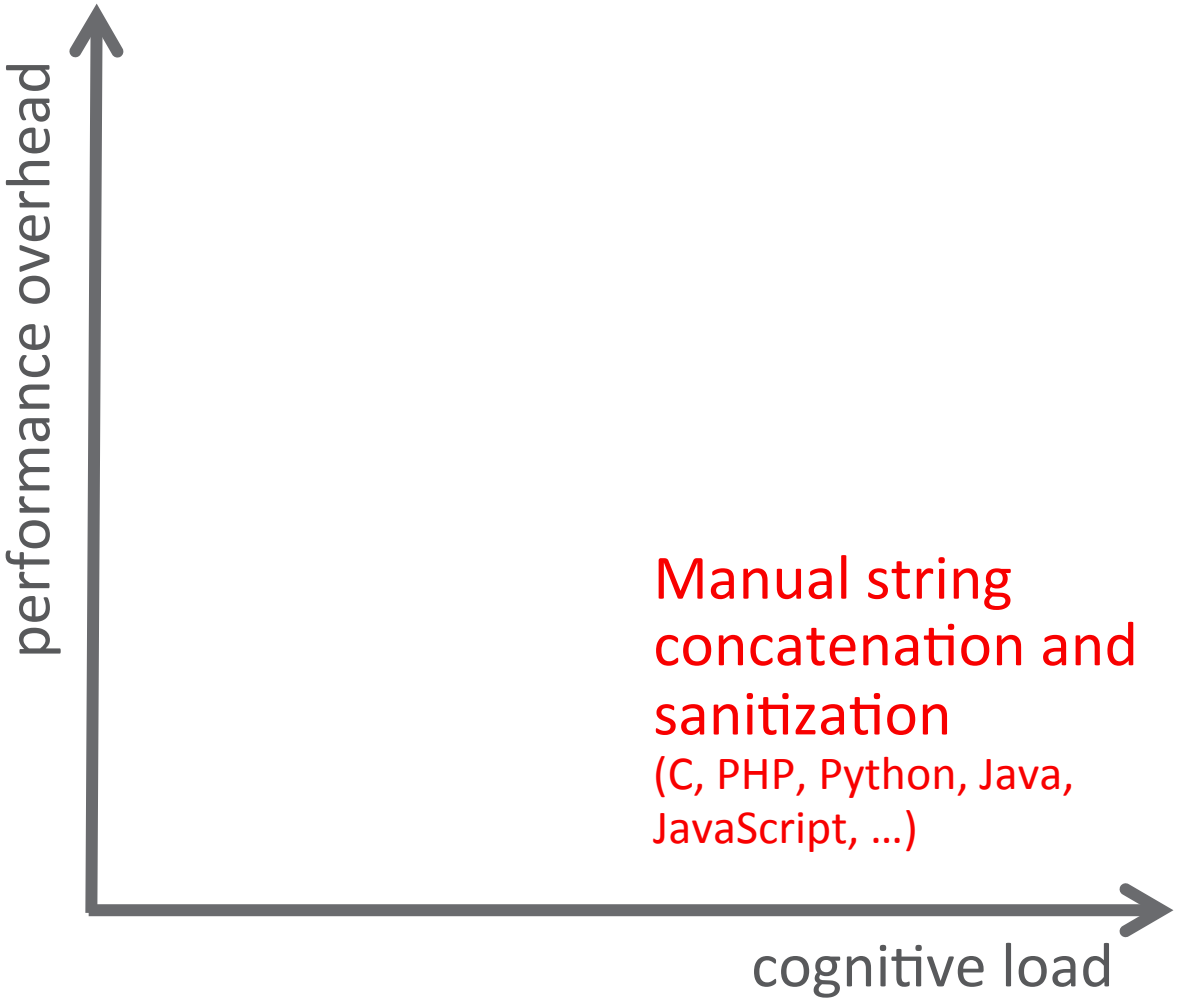
- Guaranteed memory safety
  - Ownership
  - Lifetimes
    shared borrow (`&T`)
    mutable borrow (`&mut T`)

Graydon Hoare et al, 2009

Buffer Errors – Solutions: Safe Abstractions

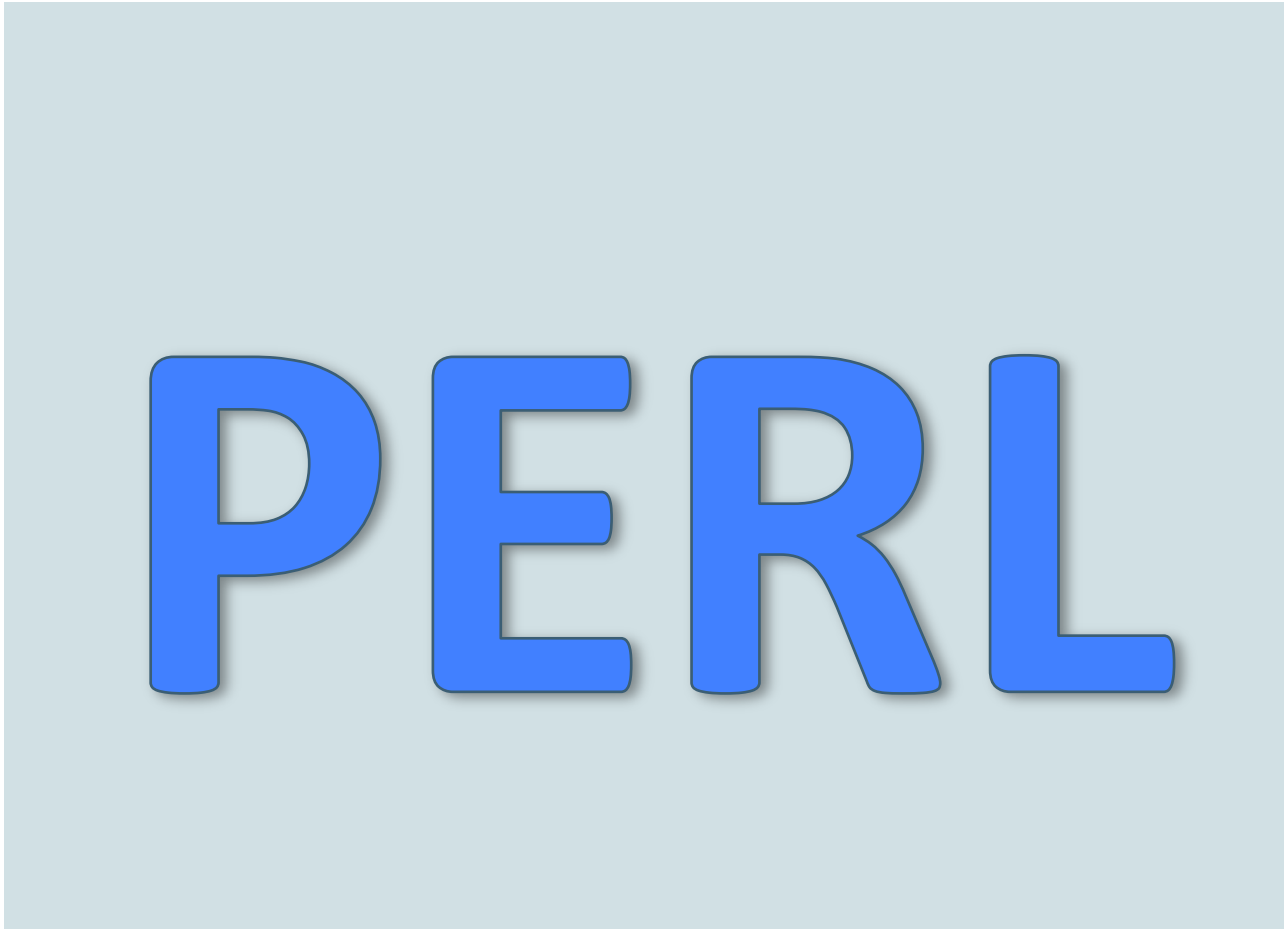


Injections – The Problem: Unsafe Abstraction



We have designed languages that prevent injections

Avoid Injections Dynamically



Larry Wall, 1987

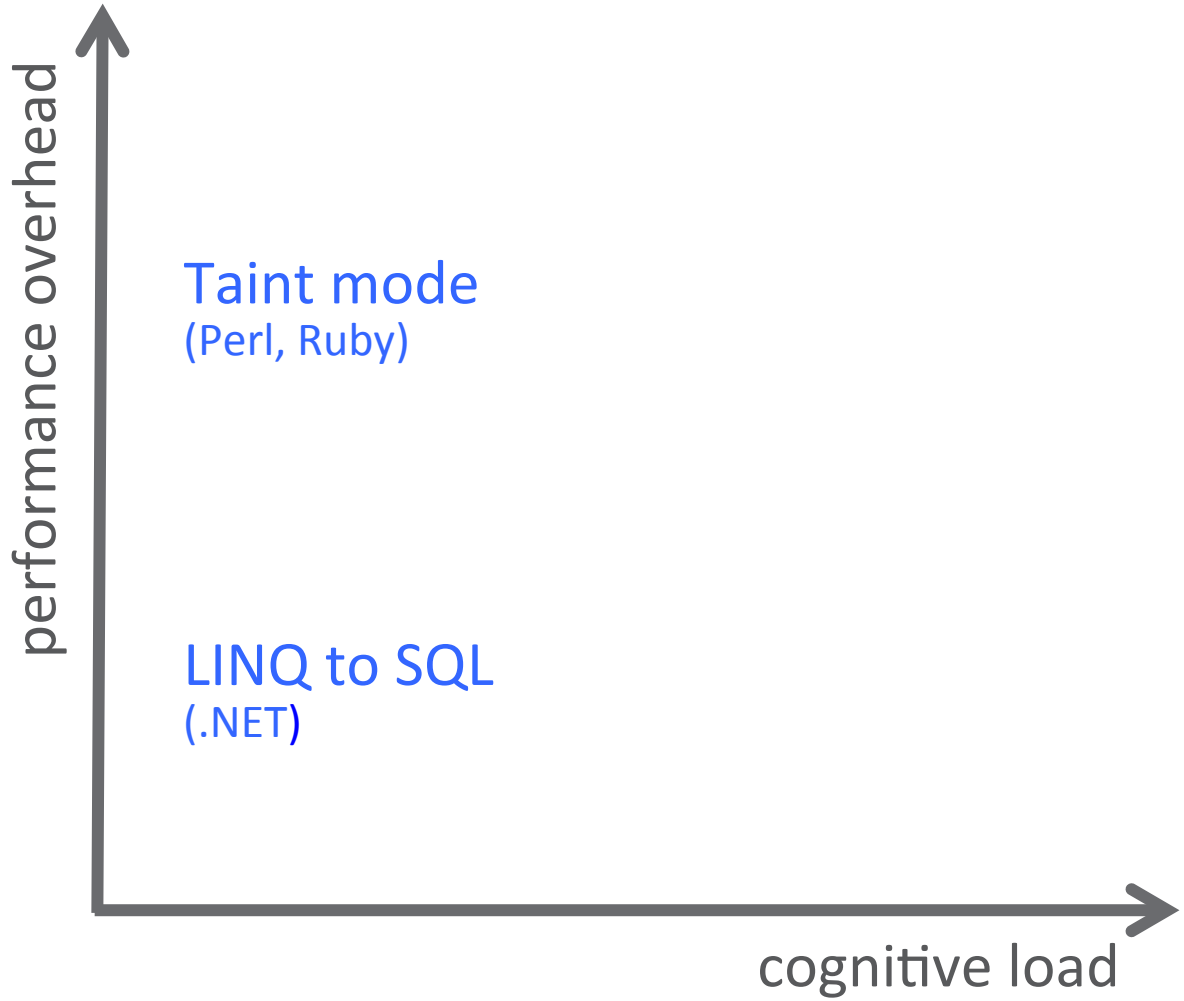
- Taint mode
  - Perl 3, 1989
  - Catches most accidental uses of untrusted string data
  - Automatic checks when program running with different real and effective user or group IDs
  - -T flag to turn it on
  - TBD for Perl 6
- Also used in
  - Ruby

Avoid SQL Injections – LINQ to SQL

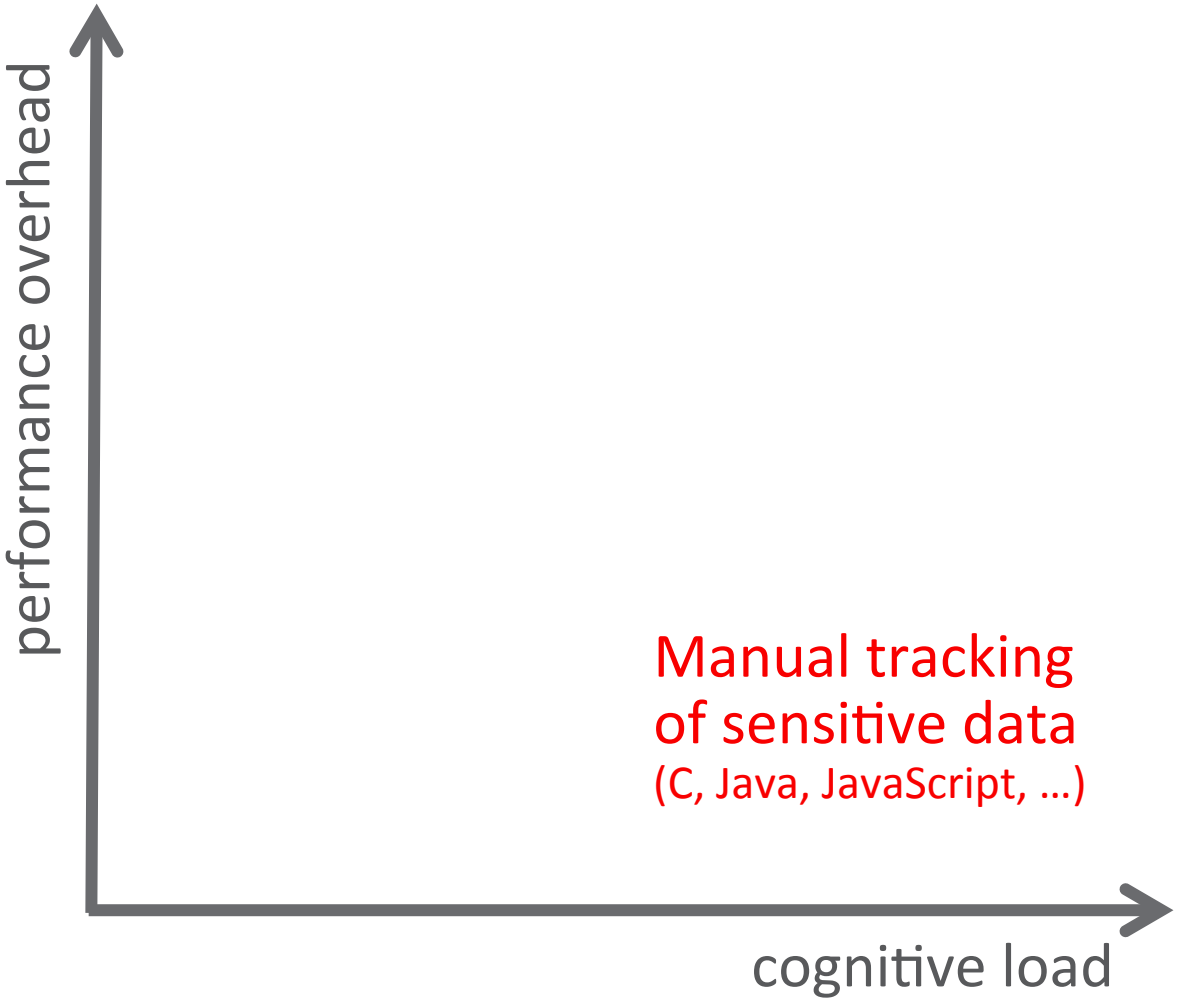
Microsoft, 2007

- .NET's Language INtegrated Query framework
- LINQ to SQL manages relational data as objects without losing the ability to query
  - Statically-typed
  - Not 100% compatible
- Avoids SQL injections by passing all data using SQL parameters
  - Not strings or string concatenation

Injections – Solutions: Safe Abstractions

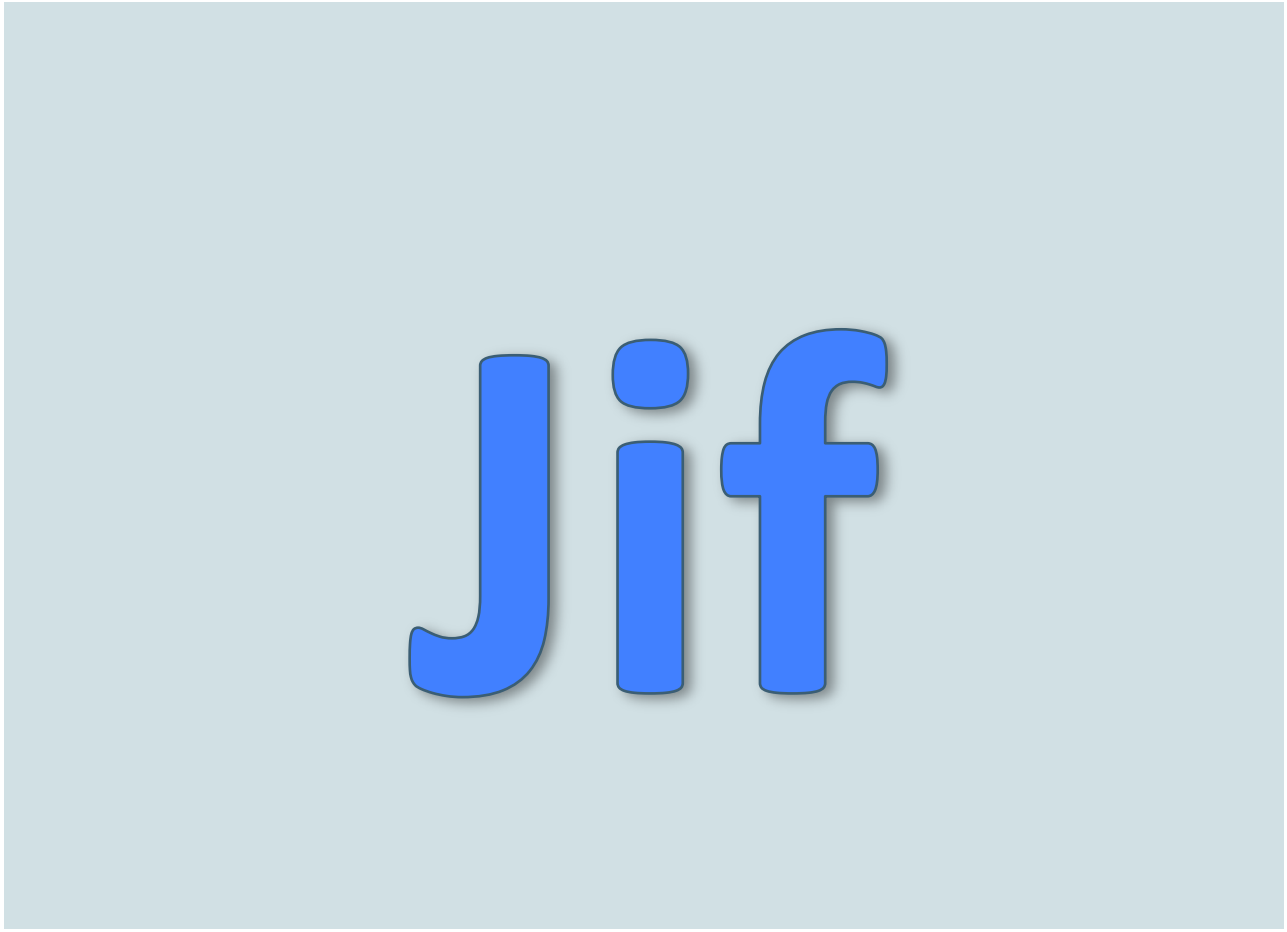


Information Leaks – The Problem: Unsafe Abstraction



We have designed academic abstractions that prevent information leaks, but they haven't made it to mainstream languages

Avoid Information Leaks and Injections Statically



- Extends Java with information flow and access control, enforced at compile time and run time
  - Integrity and confidentiality
  - Can prevent covert information leaks
- Security policies are expressed as label annotations restricting how the information may be used

Andrew Myers, 2002+

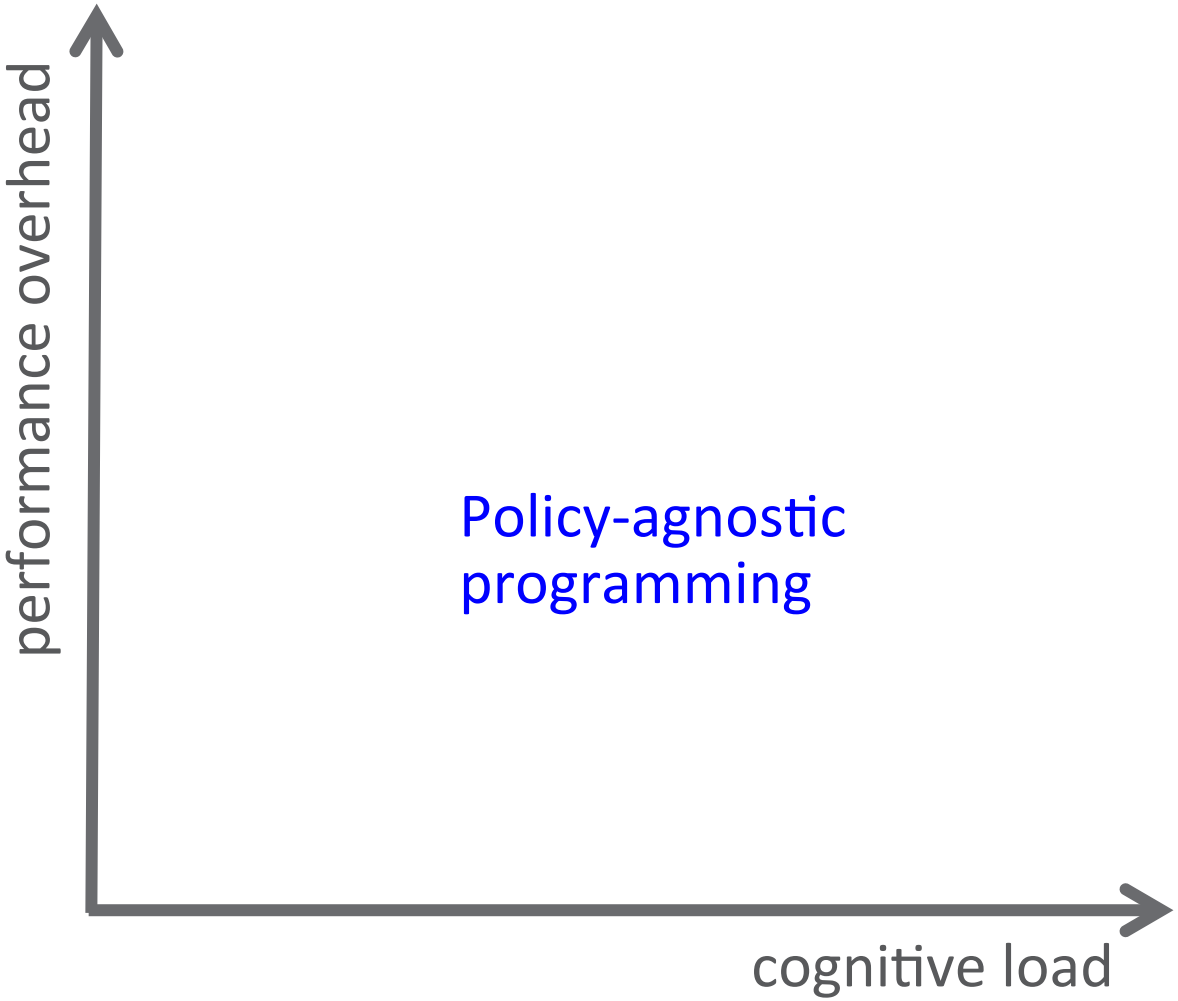
Avoid Information Leaks Dynamically

Policy-Agnostic Programming

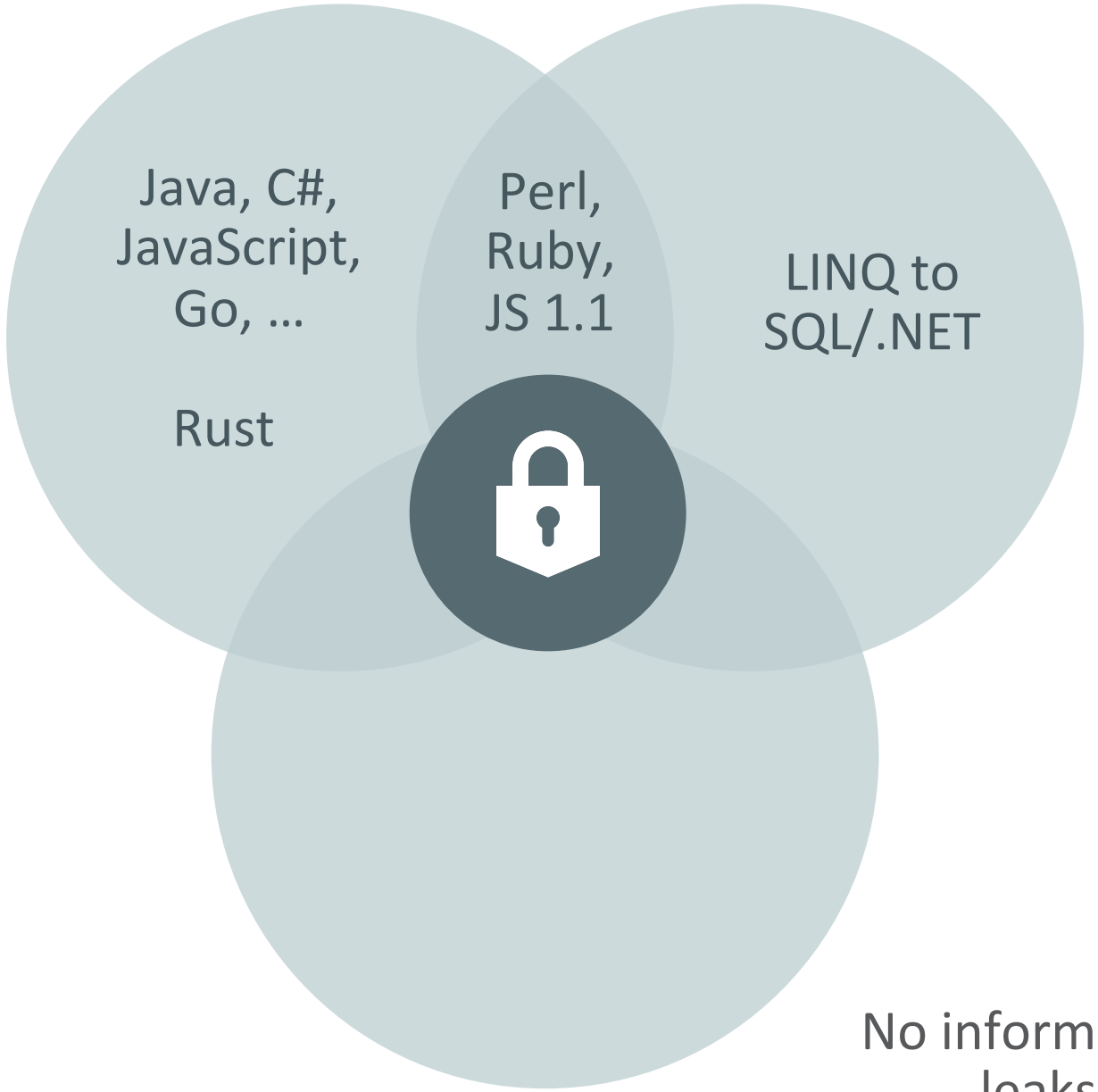
Jean Yang, 2013+

- Faceted values: a policy guarding both, the security-sensitive and non-sensitive values
  - The runtime keeps track of policies associated with conditionals
  - Faceted database saves faceted values
- Sample web applications yield reasonable (< 2x) overheads

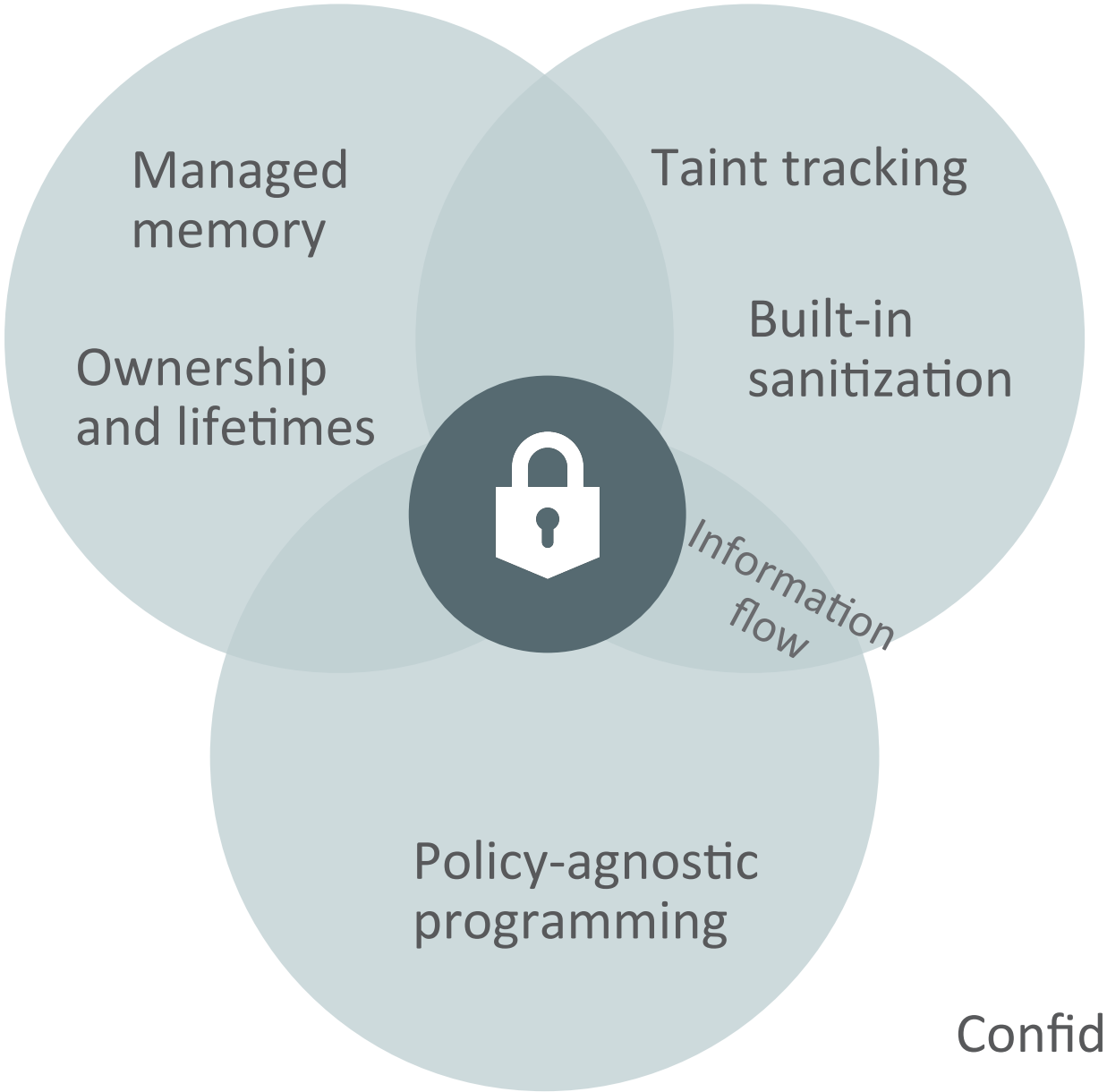
Information Leaks – Solutions: Safe Abstractions



Mainstream Languages



Abstractions



ONE LANGUAGE TO RULE THEM ALL

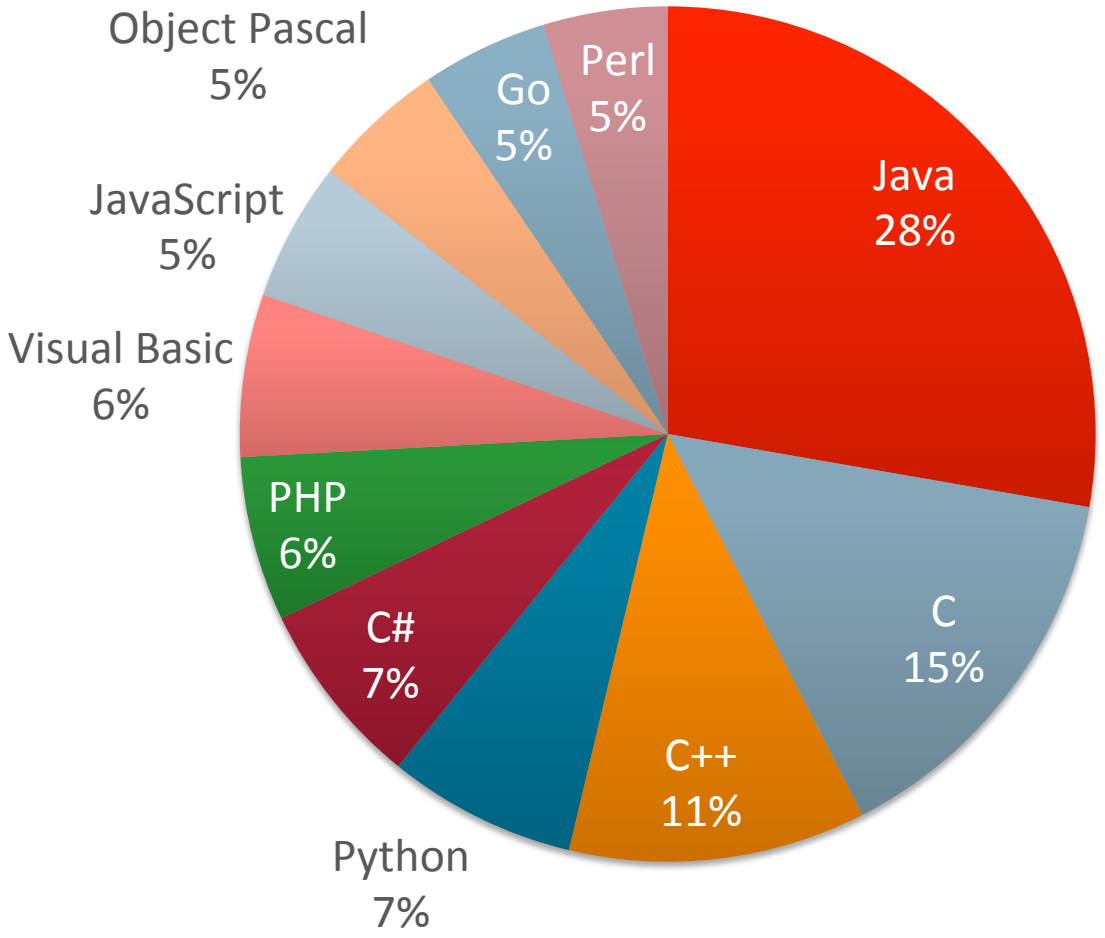

ONE
LANGUAGE
TO
RULE
THEM
ALL

27287

exploited vulnerabilities in 2013-2016

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

Top Mainstream Languages

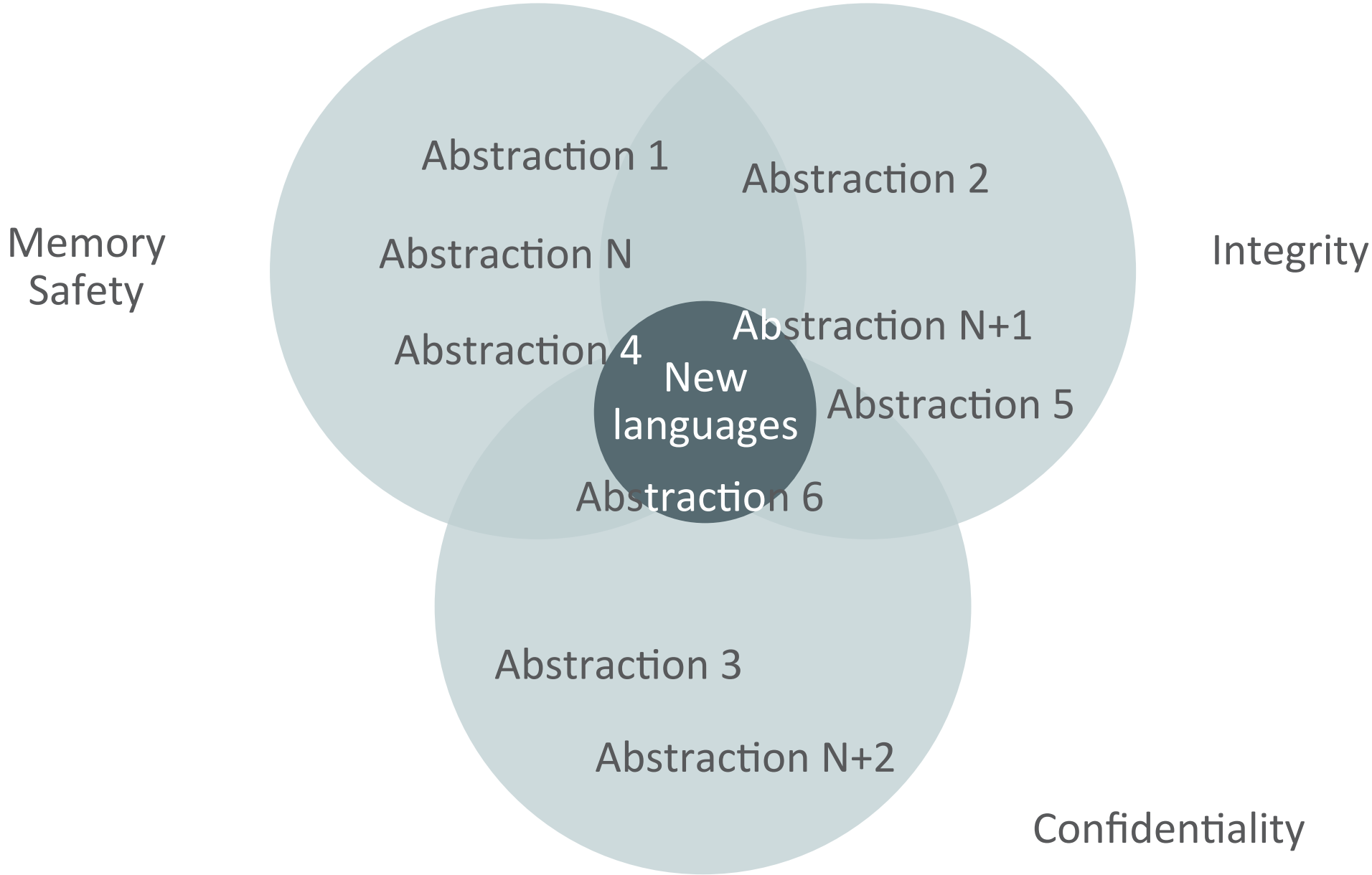


Tiobe index, July 2017

It’s time to include security in our **language design**

Challenge – To design languages that free developers from input (i.e., buffer errors, injections) and output (i.e., leaks) vulnerabilities

Challenge – To develop abstractions that minimise the cognitive load of tracking tainted data and leaked data across a system



Challenge – To provide **security guarantees** in the languages we design

18.5

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

http://www.idc.com, 2014 Worldwide Software Developer and ICT-Skilled Worker Estimations

Security is not just for expert developers


KEEP
CALM
AND
CARRY ON
PROGRAMMING

It’s time for secure languages

cristina.cifuentes@oracle.com

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

@criscifuentes

Integrated Cloud

Applications & Platform Services

Why Didn't You Mention My Favourite Mainstream Language?

Python
PHP
JavaScript
Go, …

- Provide memory safety dynamically
- Do not provide solutions for injections or information leaks


ORACLE®