Secure Programming

A.A. 2022/2023

Corso di Laurea in Ingegneria delle Telecomnicazioni

L. Python: powerful language

Paolo Ottolino

Politecnico di Bari

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE

Secure Programming Lab: Course Program

A. **Intro Secure Programming: «Who-What-Why-When-Where-How»**
B. **Building Security in: Buffer Overflow, UAF, Command Inection**
C. **SwA: Weaknesses, Vulnerabilities, Attacks**
D. **SwA (Software Assurance): Vulnerabilities and Weaknesses (CVE, OWASP, CWE)**
E. **Security & Protection: Objectives (CIA), Risks (Likelihood, Impact), Rating Methodologies**
F. **Security & Protection: Security Indicators, BIA, Protection Techniques (AAA, Listing, Duplication etc.)**
G. **Architecture and Processes: App Infrastructure, Three-Tiers, Cloud, Containers, Orchestration**
H. **Architecture and Processes 2: Ciclo di Vita del SW (SDLC), DevSecOps (OWASP DSOMM, NIST SSDF)**
I. **Free Security Tools: OWASP (ZAP, ESAPI, etc), NIST (SAMATE, SARD, SCSA, etc), SonarCube, Jenkins**
J. **Dynamic Security Test: VA, PT, DAST (cfr. VulnScanTools), WebApp Sec Scan Framework (Arachni, SCNR) :**
K. **Operating Environment: Kali Linux on WSL**
L. **Python: Powerful Language for easy creation of hacking tools**
M. **Exercises: SecureFlag**

L. Python

Agenda

L.1 Python as Hacking Tool

L.2 Energy Efficiency across Programming language

L.1 Python

Hacking Tools

Ethical Hacking with Python

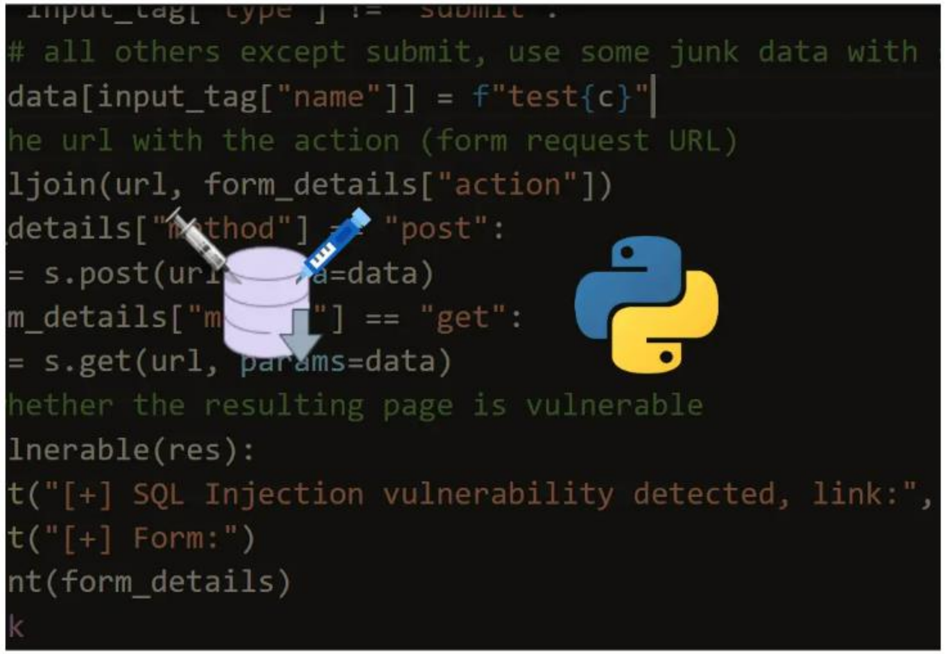
https://www.thepythoncode.com/topic/ethical-hacking

How to Inject Code into HTTP Responses in the Network in Python

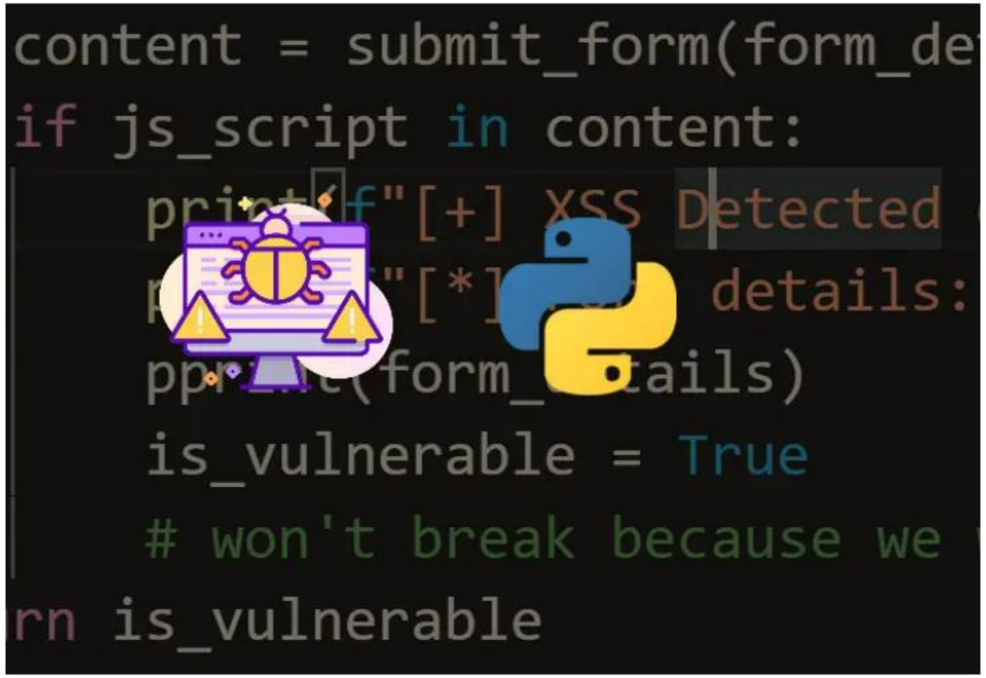
How to Make an HTTP Proxy in Python

Ethical Hacking with Python

https://www.thepythoncode.com/topic/ethical-hacking



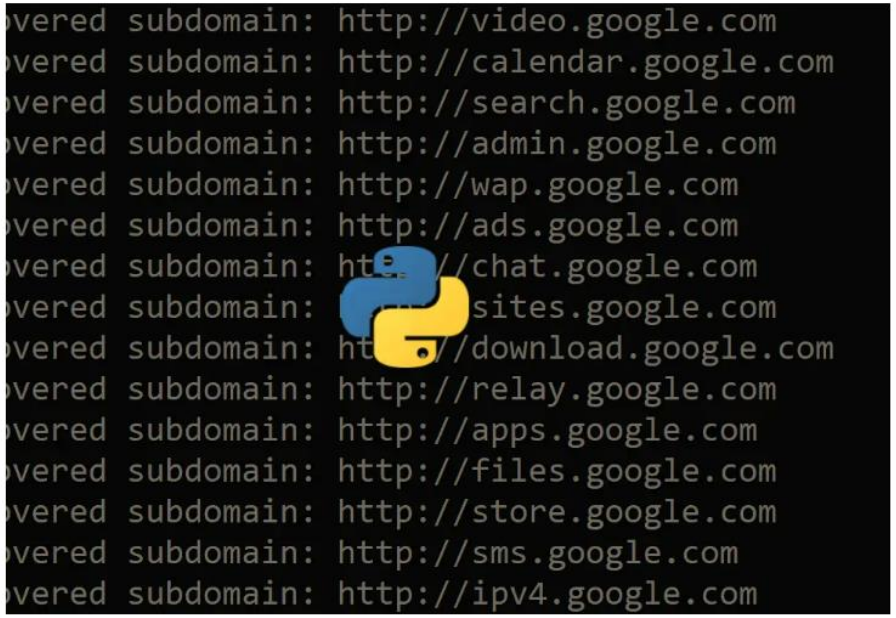
How to Build a SQL Injection Scanner in Python



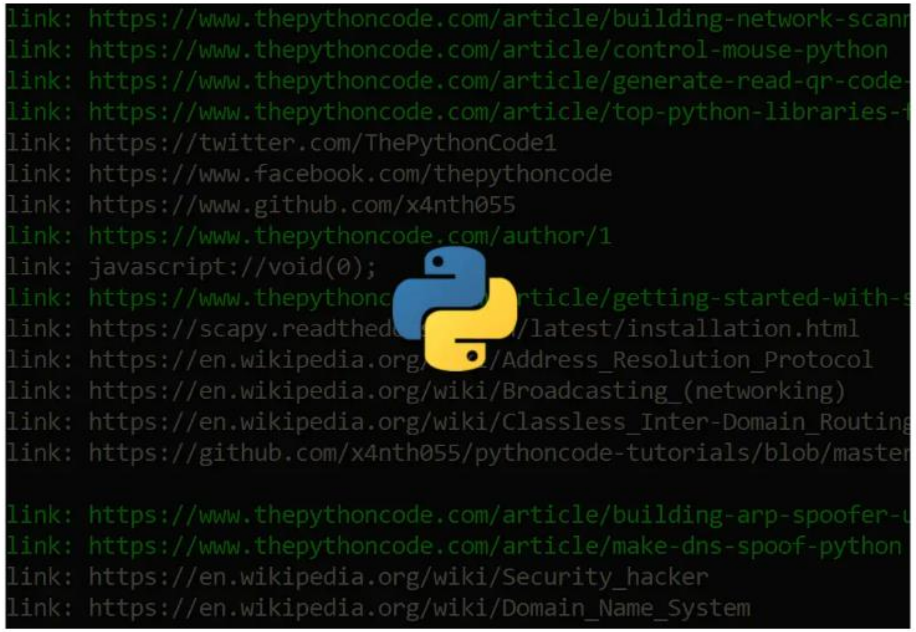
How to Build a XSS Vulnerability Scanner in Python

Ethical Hacking with Python

https://www.thepythoncode.com/topic/ethical-hacking



How to Make a Subdomain Scanner in Python



How to Extract All Website Links in Python

L.2 Energy Efficiency Across Programming Language

How Does Energy, Time, and Memory Relate

Energy Efficiency, Programming Languages, Language Benchmarking, Green Software

https://greenlab.di.uminho.pt/wp-content/uploads/2017/09/paperSLE.pdf

Energy Efficiency across Programming Languages

How Does Energy, Time, and Memory Relate?

Rui Pereira
HASLab/INESC TEC
Universidade do Minho, Portugal
ruipereira@di.uminho.pt

Marco Couto
HASLab/INESC TEC
Universidade do Minho, Portugal
marco.l.couto@inesctec.pt

Francisco Ribeiro, Rui Rua
HASLab/INESC TEC
Universidade do Minho, Portugal
fribeiro@di.uminho.pt
rrua@di.uminho.pt

Jácome Cunha
NOVA LINCS, DI, FCT
Univ. Nova de Lisboa, Portugal
jacome@fct.unl.pt

João Paulo Fernandes
Release/LISP, CISUC
Universidade de Coimbra, Portugal
jpf@dei.uc.pt

João Saraiva
HASLab/INESC TEC
Universidade do Minho, Portugal
saraiva@di.uminho.pt

Abstract

This paper presents a study of the runtime, memory usage and energy consumption of twenty seven well-known software languages. We monitor the performance of such languages using ten different programming problems, expressed in each of the languages. Our results show interesting findings, such as, slower/faster languages consuming less/more energy, and how memory usage influences energy consumption. We show how to use our results to provide software engineers support to decide which language to use when energy efficiency is a concern.

CCS Concepts • **Software and its engineering → Software performance; General programming languages;**

Keywords Energy Efficiency, Programming Languages, Language Benchmarking, Green Software

ACM Reference format:
Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. 2017. Energy Efficiency across Programming Languages. In *Proceedings of SLE'17, Vancouver, BC, Canada, October 23–24, 2017,* 12 pages.
https://doi.org/10.1145/3136014.3136031

productivity - by incorporating advanced features in the language design, like for instance powerful modular and type systems - and at efficiently execute such software - by developing, for example, aggressive compiler optimizations. Indeed, most techniques were developed with the main goal of helping software developers in producing faster programs. In fact, in the last century *performance* in software languages was in almost all cases synonymous of *fast execution time* (embedded systems were probably the single exception).

In this century, this reality is quickly changing and software energy consumption is becoming a key concern for computer manufacturers, software language engineers, programmers, and even regular computer users. Nowadays, it is usual to see mobile phone users (which are powerful computers) avoiding using CPU intensive applications just to save battery/energy. While the concern on the computers' energy efficiency started by the hardware manufacturers, it quickly became a concern for software developers too [27]. In fact, this is a recent and intensive area of research where several techniques to analyze and optimize the energy consumption of software systems are being developed. Such techniques already provide knowledge on the energy effi-

Software Language Engineering

design, implement and evolve software languages:

Why	Who/What	How	When	Where
Improving Productivity	Programmers	incorporating advanced features in the language design	Developing: powerful modular and type systems	This century
Efficiencing Execution	Developed Software	Reducing memory consumption, instructions	Runtime: aggressive compiler optimizations	Last century

… but the reality is quickly changing and **software energy consumption** is becoming a **key concern**

Software Language Engineering

design, implement and evolve software languages:

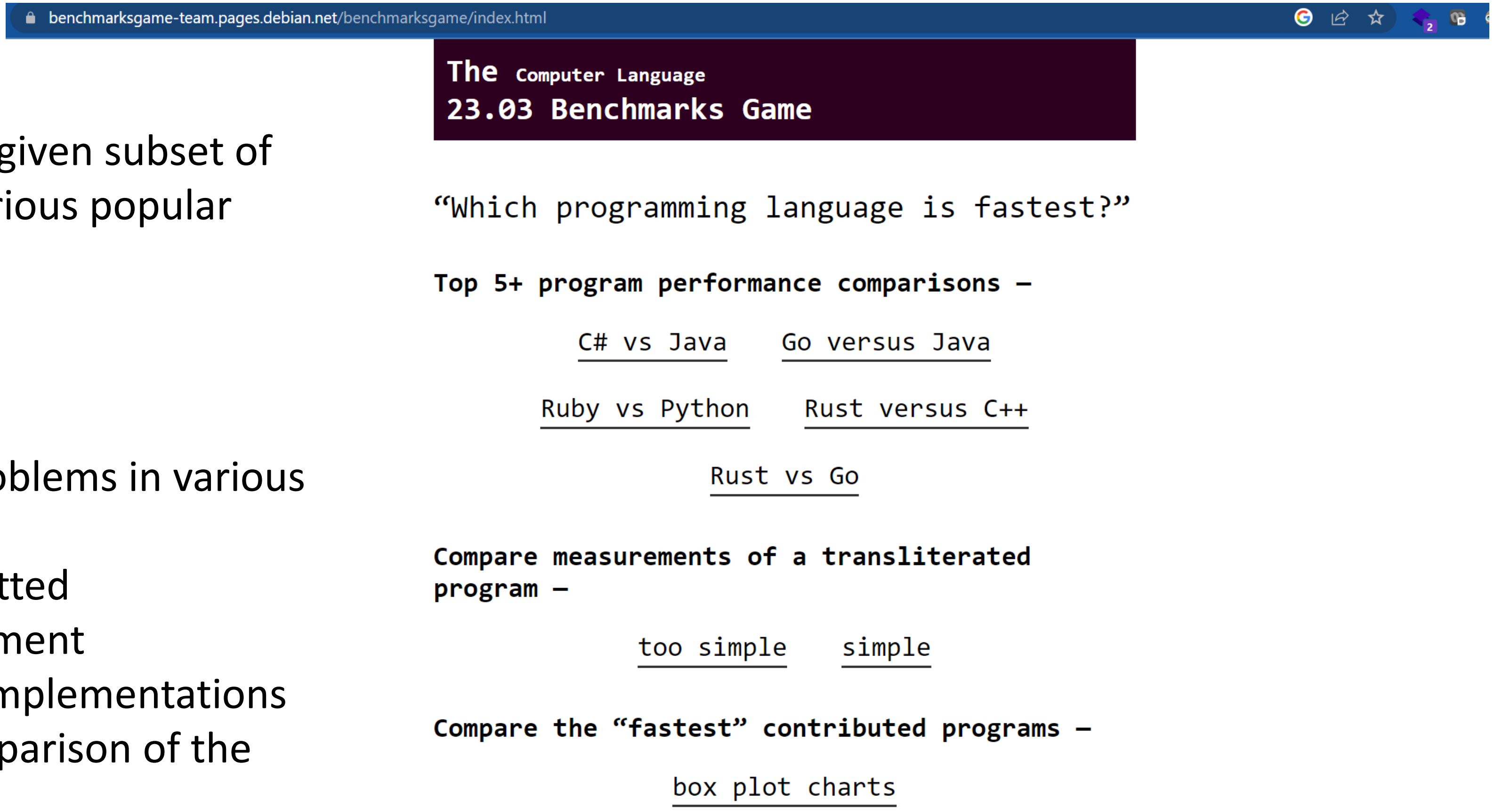
Why	Who/What	How	When	Where
Improving Productivity	Programmers	incorporating advanced features in the language design	Developing: powerful modular and type systems	This century
Efficiencing Execution	Developed Software	Reducing memory consumption, instructions	Runtime: aggressive compiler optimizations	Last century

… but the reality is quickly changing and **software energy consumption** is becoming a **key concern**

L.2c Energy Efficiency Across Programming Language

How Does Energy, Time, and Memory Relate

CLBG: Computer Language Benchmarks Game



a free software project for comparing how a given subset of simple algorithms can be implemented in various popular programming languages.

The project consists of:

- A set of very simple algorithmic problems
- Various implementations to the above problems in various programming languages
- A set of unit tests to verify that the submitted implementations solve the problem statement
- A framework for running and timing the implementations
- A website to facilitate the interactive comparison of the results

L.2d Energy Efficiency Across Programming Language

Test Design and Execution (based on CLBG)

Benchmark	Description	Input
n-body	Double precision N-body simulation	50M
fannkuch-redux	Indexed access to tiny integer sequence	12
spectral-norm	Eigenvalue using the power method	5,500
mandelbrot	Generate Mandelbrot set portable bitmap file	16,000
pidigits	Streaming arbitrary precision arithmetic	10,000
regex-redux	Match DNA 8mers and substitute magic patterns	fasta output
fasta	Generate and write random DNA sequences	25M
k-nucleotide	Hashtable update and k-nucleotide strings	fasta output
reverse-complement	Read DNA sequences, write their reverse-complement	fasta output
binary-trees	Allocate, traverse and deallocate many binary trees	21
chameneos-redux	Symmetrical thread rendezvous requests	6M
meteor-contest	Search for solutions to shape packing puzzle	2,098
thread-ring	Switch from thread to thread passing one token	50M

Suite of programs

Paradigm	Languages
Functional	Erlang, F#, Haskell, Lisp, Ocaml, Perl, Racket, Ruby, Rust;
Imperative	Ada, C, C++, F#, Fortran, Go, Ocaml, Pascal, Rust;
Object-Oriented	Ada, C++, C#, Chapel, Dart , F#, Java, JavaScript, Ocaml, Perl, PHP, Python, Racket, Rust, Smalltalk, Swift, TypeScript;
Scripting	Dart, Hack, JavaScript, JRuby, Lua, Perl, PHP, Python, Ruby, TypeScript;

Suite of languages
(having free available compilers → 27)

➔ **The Energy Consumption**
Is calculated by the energy consuption of the system call (RAPL: Running Average Power Limit)

L.2e Energy Efficiency Across Programming Language

Results

The greenest Language is C

Python requires about **76** time the **energy** of **C** for executing the same activity

Python takes about **72** more **time** than **C** for executing the same activities

Python asks for about **2-3** times the size of **memory** required by **P** for executing the same activities (Java 6)

Total

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84