Course Information

NET7212 — Safe System Programming (in Rust)

Samuel Tardieu   Stefano Zacchiroli
2023-09-18

Objectives

What you will learn

In this course you will learn how to build **system-level applications** that avoid by construction **memory safety issues** and **data race issues**, by relying on modern type systems.

You will be introduced to **Rust** as an example of a programming language that realizes this approach and has significant industry adoption.

Prerequisites

- Fundamentals of operating systems
- C programming (C++ would be a plus)
- POSIX programming
- Some experience with multithreading/multiprocessing programming

Syllabus

- Memory safety
- How to detect memory-safety issues in C/C++
- The Rust memory model
- NULL references and how to avoid "billion dollar mistakes"
- Rust language basics
- Race conditions
- Avoiding multiprocessing (security) pitfalls
- Data races
- Avoiding multithreading (security) pitfalls

2022 CWE Top 25 — redux

Rank	ID	Name	Score	KEV Count (CVEs)	Rank Change vs. 2021
1	CWE-787	Out-of-bounds Write	64.20	62	0
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.97	2	0
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	22.11	7	+3 ▲
4	CWE-20	Improper Input Validation	20.63	20	0
5	CWE-125	Out-of-bounds Read	17.67	1	-2 ▼
6	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	17.53	32	-1 ▼
7	CWE-416	Use After Free	15.50	28	0
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.08	19	0
9	CWE-352	Cross-Site Request Forgery (CSRF)	11.53	1	0
10	CWE-434	Unrestricted Upload of File with Dangerous Type	9.56	6	0
11	CWE-476	NULL Pointer Dereference	7.15	0	+4 ▲
12	CWE-502	Deserialization of Untrusted Data	6.68	7	+1 ▲
13	CWE-190	Integer Overflow or Wraparound	6.53	2	-1 ▼
14	CWE-287	Improper Authentication	6.35	4	0
15	CWE-798	Use of Hard-coded Credentials	5.66	0	+1 ▲
16	CWE-862	Missing Authorization	5.53	1	+2 ▲
17	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	5.42	5	+8 ▲
18	CWE-306	Missing Authentication for Critical Function	5.15	6	-7 ▼
19	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.85	6	-2 ▼
20	CWE-276	Incorrect Default Permissions	4.84	0	-1 ▼
21	CWE-918	Server-Side Request Forgery (SSRF)	4.27	8	+3 ▲
22	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	3.57	6	+11 ▲
23	CWE-400	Uncontrolled Resource Consumption	3.56	2	+4 ▲
24	CWE-611	Improper Restriction of XML External Entity Reference	3.38	0	-1 ▼
25	CWE-94	Improper Control of Generation of Code ('Code Injection')	3.32	4	+3 ▲

2022 CWE Top 25 vs this course

Rank	ID	Name	Score	KEV Count (CVEs)	Rank Change vs. 2021
1	CWE-787	Out-of-bounds Write	64.20	62	0
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.97	2	0
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	22.11	7	+3 ▲
4	CWE-20	Improper Input Validation	20.63	20	0
5	CWE-125	Out-of-bounds Read	17.67	1	-2 ▼
6	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	17.53	32	-1 ▼
7	CWE-416	Use After Free	15.50	28	0
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.08	19	0
9	CWE-352	Cross-Site Request Forgery (CSRF)	11.53	1	0
10	CWE-434	Unrestricted Upload of File with Dangerous Type	9.56	6	0
11	CWE-476	NULL Pointer Dereference	7.15	0	+4 ▲
12	CWE-502	Deserialization of Untrusted Data	6.68	7	+1 ▲
13	CWE-190	Integer Overflow or Wraparound	6.53	2	-1 ▼
14	CWE-287	Improper Authentication	6.35	4	0
15	CWE-798	Use of Hard-coded Credentials	5.66	0	+1 ▲
16	CWE-862	Missing Authorization	5.53	1	+2 ▲
17	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	5.42	5	+8 ▲
18	CWE-306	Missing Authentication for Critical Function	5.15	6	-7 ▼
19	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.85	6	-2 ▼
20	CWE-276	Incorrect Default Permissions	4.84	0	-1 ▼
21	CWE-918	Server-Side Request Forgery (SSRF)	4.27	8	+3 ▲
22	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	3.57	6	+11 ▲
23	CWE-400	Uncontrolled Resource Consumption	3.56	2	+4 ▲
24	CWE-611	Improper Restriction of XML External Entity Reference	3.38	0	-1 ▼
25	CWE-94	Improper Control of Generation of Code ('Code Injection')	3.32	4	+3 ▲

You will learn how to avoid *by design* the CWEs highlighted in red using modern type systems and programming language technology. (And, to a lesser extent, how to avoid or mitigate the CWE highlighted in orange, and more.)

Team

Samuel Tardieu ⊡

- Associate Professor at Télécom Paris, IP Paris, team ACES
- Research: operating systems and schedulers, distributed systems, embedded systems and robotics
- Free software enthusiast and activist for 30 years
- Programming languages polyglot, with a preference for Rust, Haskell, Scala, Factor, and Scheme

Stefano Zacchiroli ⊡

- Full Professor at Télécom Paris, IP Paris, team ACES
- Research: free/open source software (FOSS) software engineering, digital commons, cybersecurity, software supply chain
- Co-founder and CTO of Software Heritage ⊡, the largest public archive of FOSS source code
- Tech and activism: Debian ⊡ developer, former Debian project leader (2010-2013), former board director of the Open Source Initiative (OSI) ⊡

Evaluation

Final grade

50% written exam + 50% project assignment

Project assignment

- General idea: implement in Rust a secure and fast system application for a real-world use case.
- Project to be implemented either solo or in pairs.
- Evaluation criteria: specification conformity, performances, robustness (including security), code quality.
- Evaluation methods: code review, testing, discussion of design/implementation choices with the teachers.

(Project presentation to follow.)

Organization

- **Weekly lectures** on Tuesday, usually (with exceptions):

  - 1 TH of lecture (*cours magistral*)
  - 1 TH of lab work (*TP*)

  Teacher will vary (S. Tardieu / S. Zacchiroli)

- **Personal or group work** on the project: on your own time.

  Try to allocate at least 3 hours/week to the project, otherwise you won't make it.

Contacts

Practical information

→ on Synapses ⤤

Email

- For *everything* about the course: ‹net7212-2324@listes.telecom-paris.fr› ⤤ mailing-list
  - Both students and teachers are subscribed
  - Teachers will answer on-list to inquiries posted there by anyone, so that everybody benefit
  - Mutual help among students on-list is encouraged
- For private inquiries only (e.g., "I'm sick"):
  - Samuel Tardieu ‹samuel.tardieu@telecom-paris.fr› ⤤
  - Stefano Zacchiroli ‹stefano.zacchiroli@telecom-paris.fr› ⤤

  Warning: private email inquiries that could have been asked via the mailing list will be ignored.

Course material

Homepage

https://net7212.telecom-paris.fr/

Sub-pages linked from homepage for:

- Lecture slides (PDF)
  - Published *after* each lecture
- Lab assignments (HTML)
  - Published *before* each lab session
- Project specification (HTML)
  - First version available now; will be updated during the teaching period with more milestones

Good reads

System programming

1. Robert Love. Linux System Programming: Talking Directly to the Kernel and C Library ⧉, 2nd edition. O'Reilly Media, 2013.
2. Michael Kerrisk. The Linux Programming Interface: A Linux and UNIX System Programming Handbook ⧉. No Starch Press, 2010.

Rust programming

1. Steve Klabnik, Carol Nichols. The Rust Programming Language ⧉, 2nd edition. No Starch Press, 2023.
2. Jim Blandy, Jason Orendorff, Leonora F. S. Tindall. Programming Rust, 2nd Edition ⧉. O'Reilly Media, Inc. 2021.
3. Rust community, The Rustonomicon ⧉: The Dark Arts of Unsafe Rust

The floor is yours!

?