MITRE 2023 CWE Top 25

cwe.mitre.org/top25/archive/2023/2023_top25_list.html#tableView

Most Dangerous
Software Weaknesses

1	**Out-of-bounds Write**	63.72
2	Improper Neutralization of Input During Web Page Gen. (Cross-site Scripting)	45.54
3	Improper Neutralization of Special Elements used in … (SQL Injection)	34.27
4	**Use After Free**	16.71
5	Improper Neutralization of Special Elements used in … (OS Cmd Injection)	15.65
6	Improper Input Validation	15.5
7	**Out-of-bounds Read**	14.6
8	Improper Limitation … to a Restricted Directory (Path Traversal)	14.11
9	Cross-Site Request Forgery (CSRF)	11.73
10	Unrestricted Upload of File with Dangerous Type	10.41

What "is" C++'s language safety problem (2)

C++ should provide a way to let programmers

- **by default enforce** known rules in these areas, with explicit opt-out
- **aiming for a ~90-98% reduction** in these vulnerabilities (parity with other langs)

But right away let's clarify, and set some boundaries:

- "Immediate": The start, **not the end** (e.g., let's improve concurrency safety too)
- "Default" + "enforcement": Need a mode where "if it compiles, it's in the safe subset unless you explicitly opt out" (aka **bright line**)
- "Known rules": A great start, but also have a few **gaps to fill** (esp. bounds checking)
- "~90-98% improvement": That can be achieved with **full compatibility**, but trying for 100% is a mistake (not necessary for parity, not sufficient, and breaking compatibility would be too high a cost)