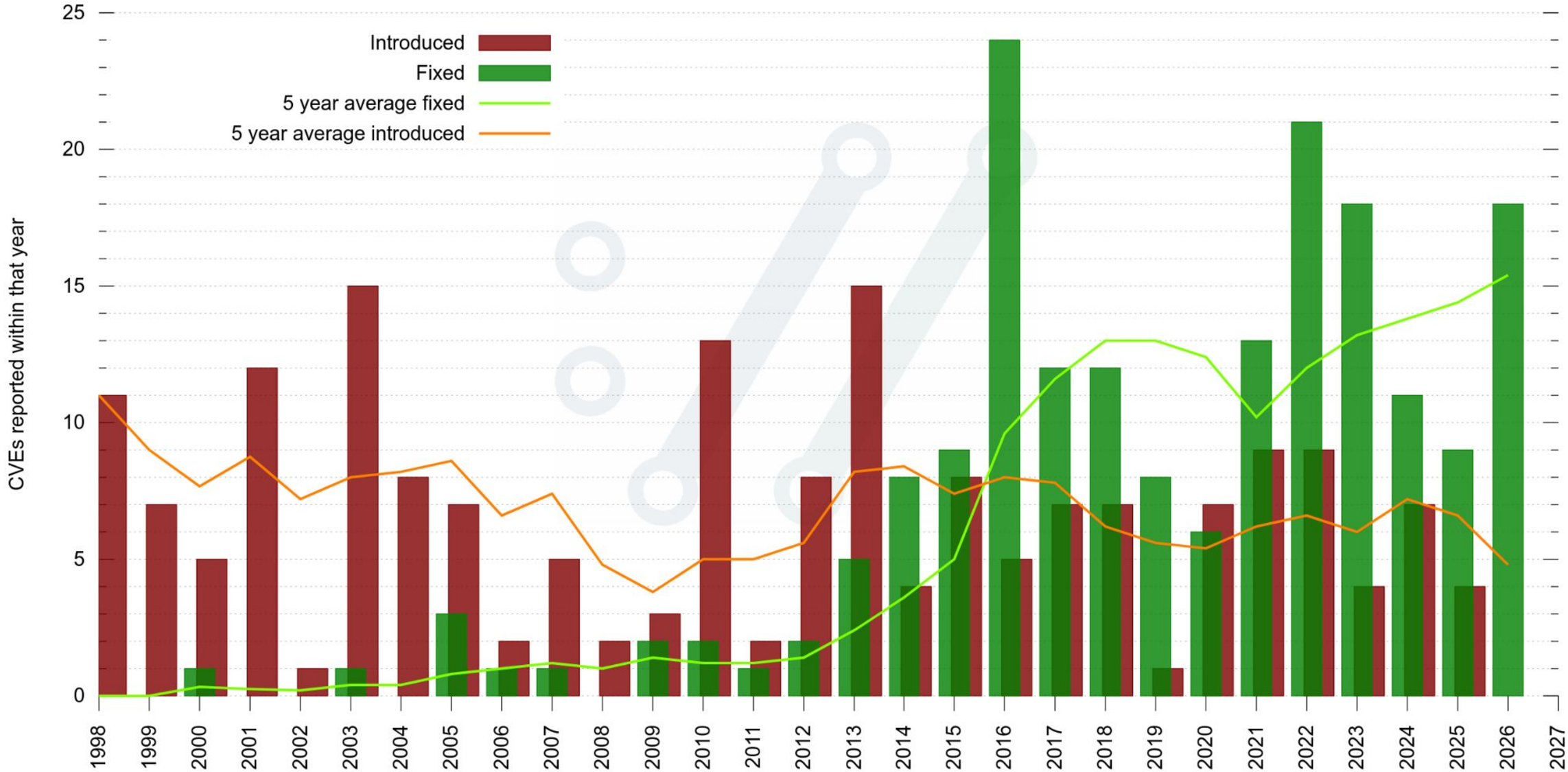
curl://up
2026

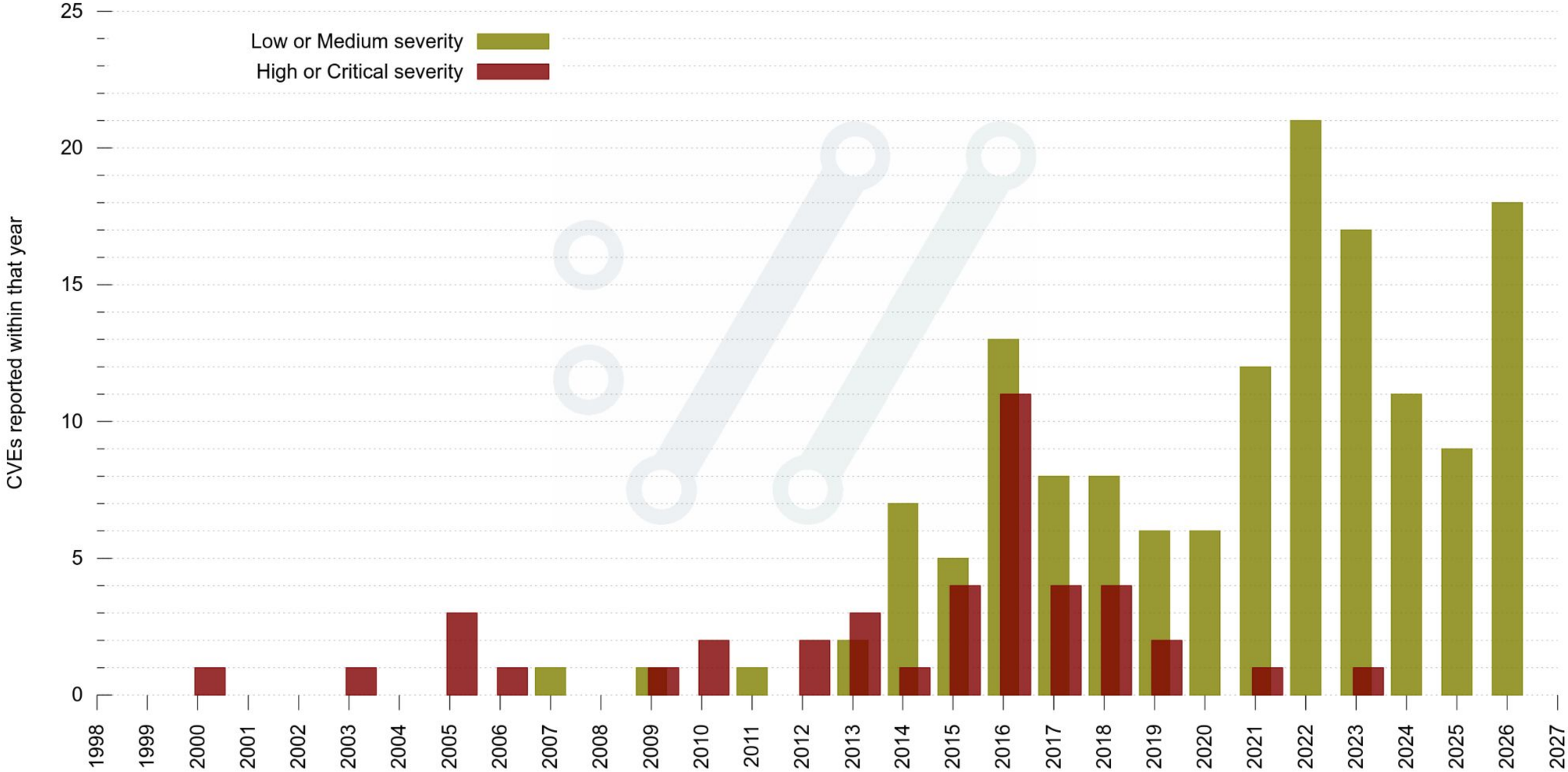
security

trends

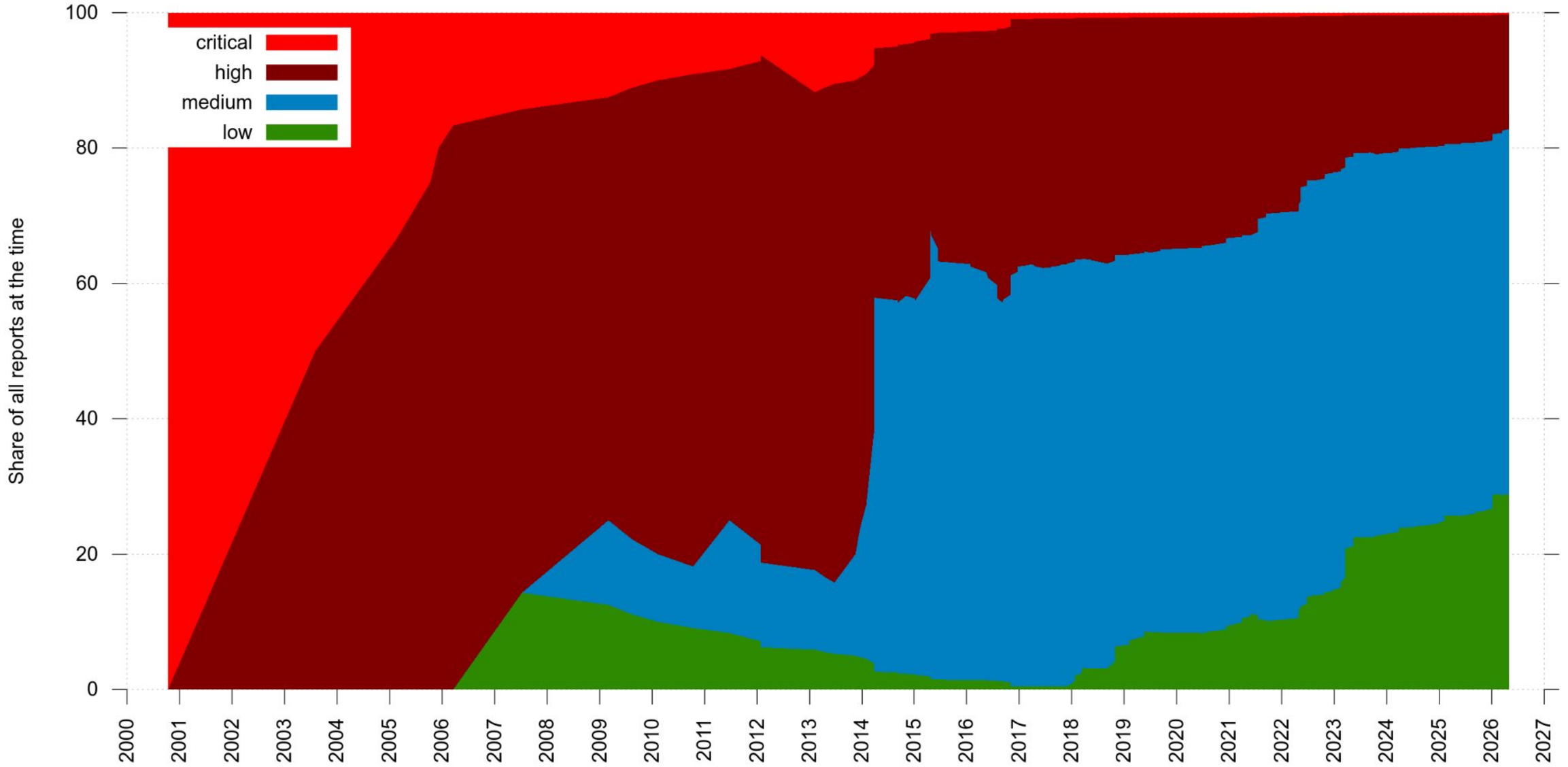
Vulnerabilities Fixed/Introduced



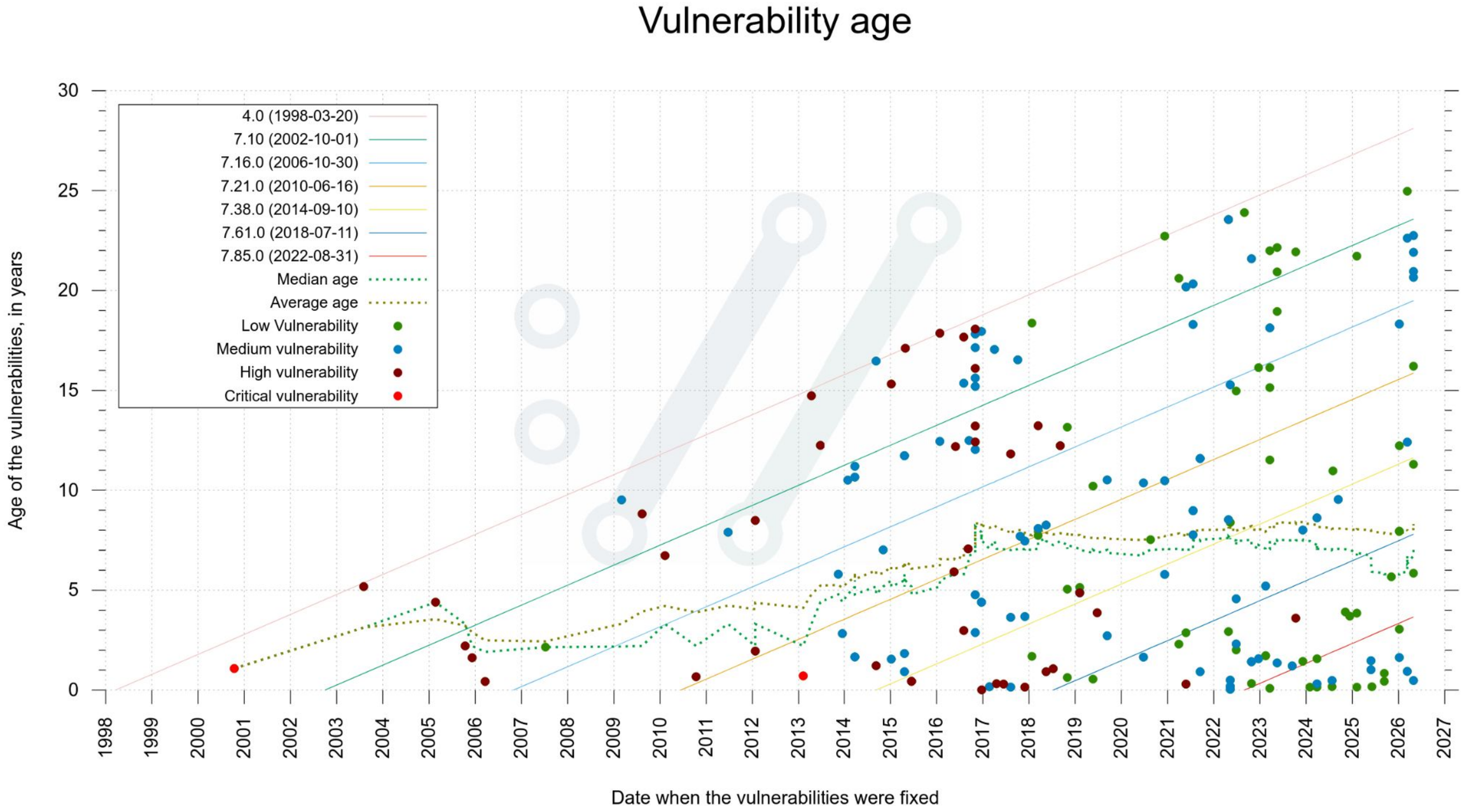
Vulnerability reports high vs low



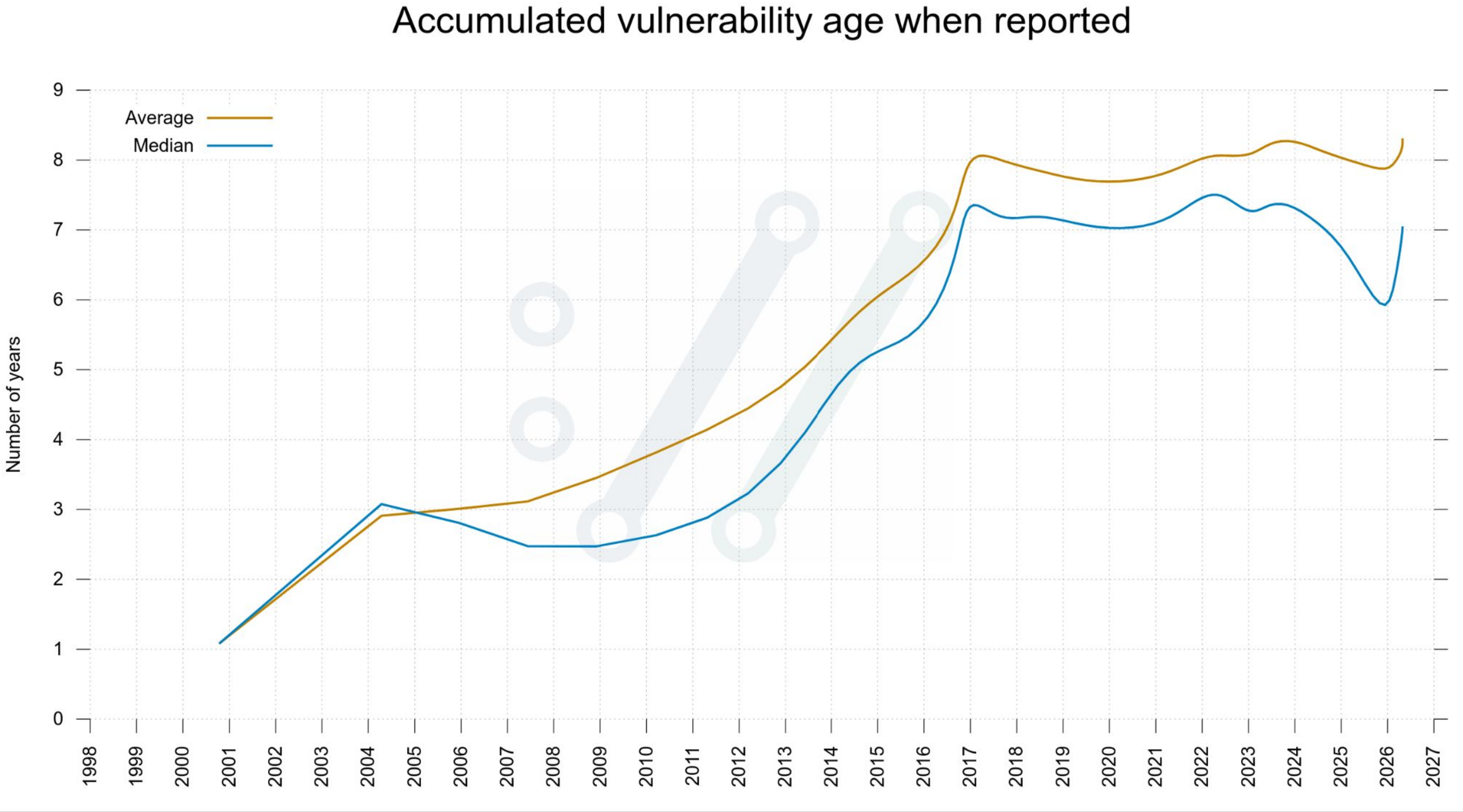
Severity distribution among all curl vulnerability reports accumulated



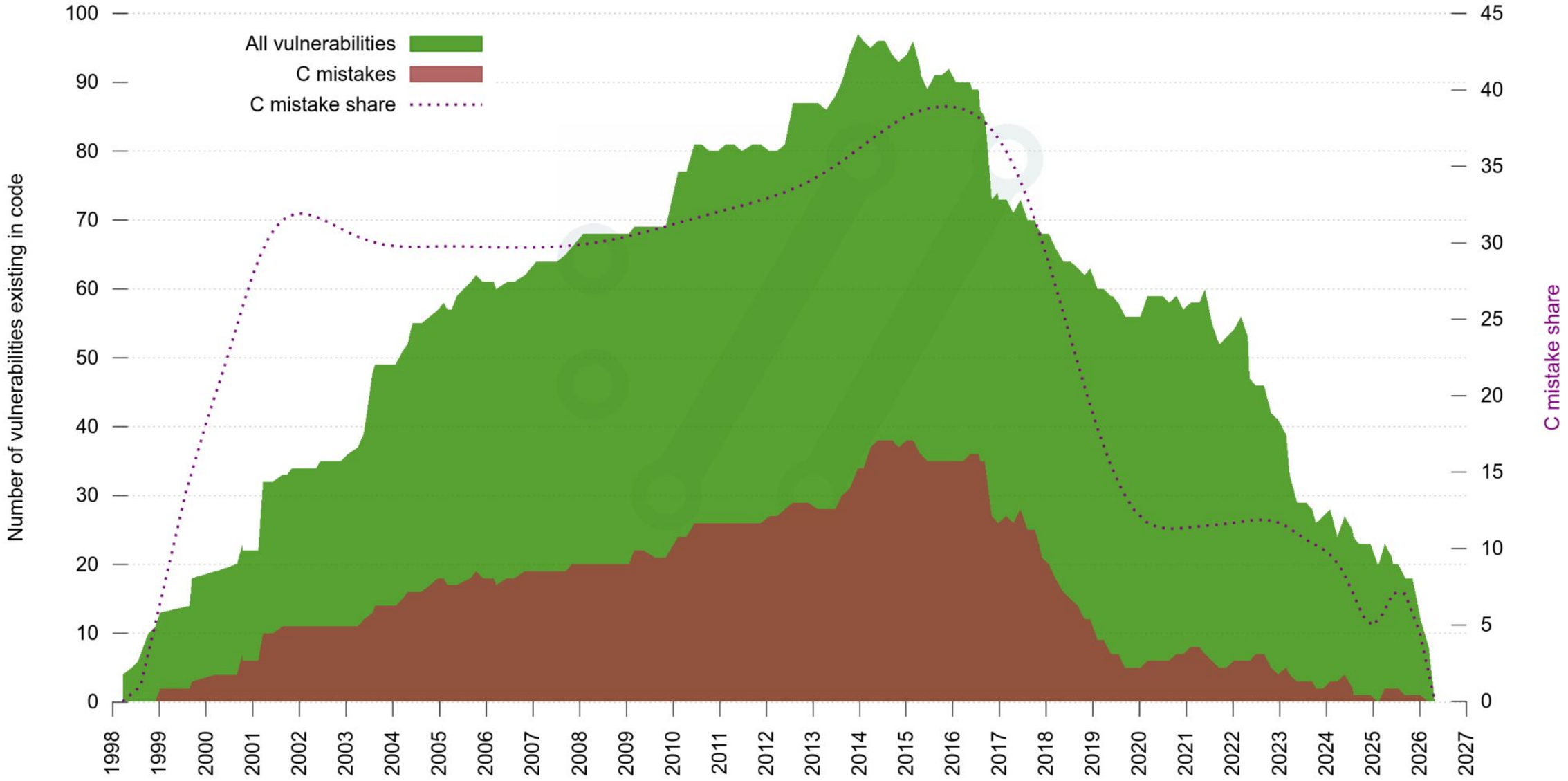

Time from security report to shipped fix
Individual flaw
All-time average
12-month average
Number of days
110
100
90
80
70
60
50
40
30
20
10
0
All vulnerabilities ever published


Vulnerability age
Age of the vulnerabilities, in years
Date when the vulnerabilities were fixed
4.0 (1998-03-20)
7.10 (2002-10-01)
7.16.0 (2006-10-30)
7.21.0 (2010-06-16)
7.38.0 (2014-09-10)
7.61.0 (2018-07-11)
7.85.0 (2022-08-31)
Median age
Average age
Low Vulnerability
Medium vulnerability
High vulnerability
Critical vulnerability
0
5
10
15
20
25
30
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027

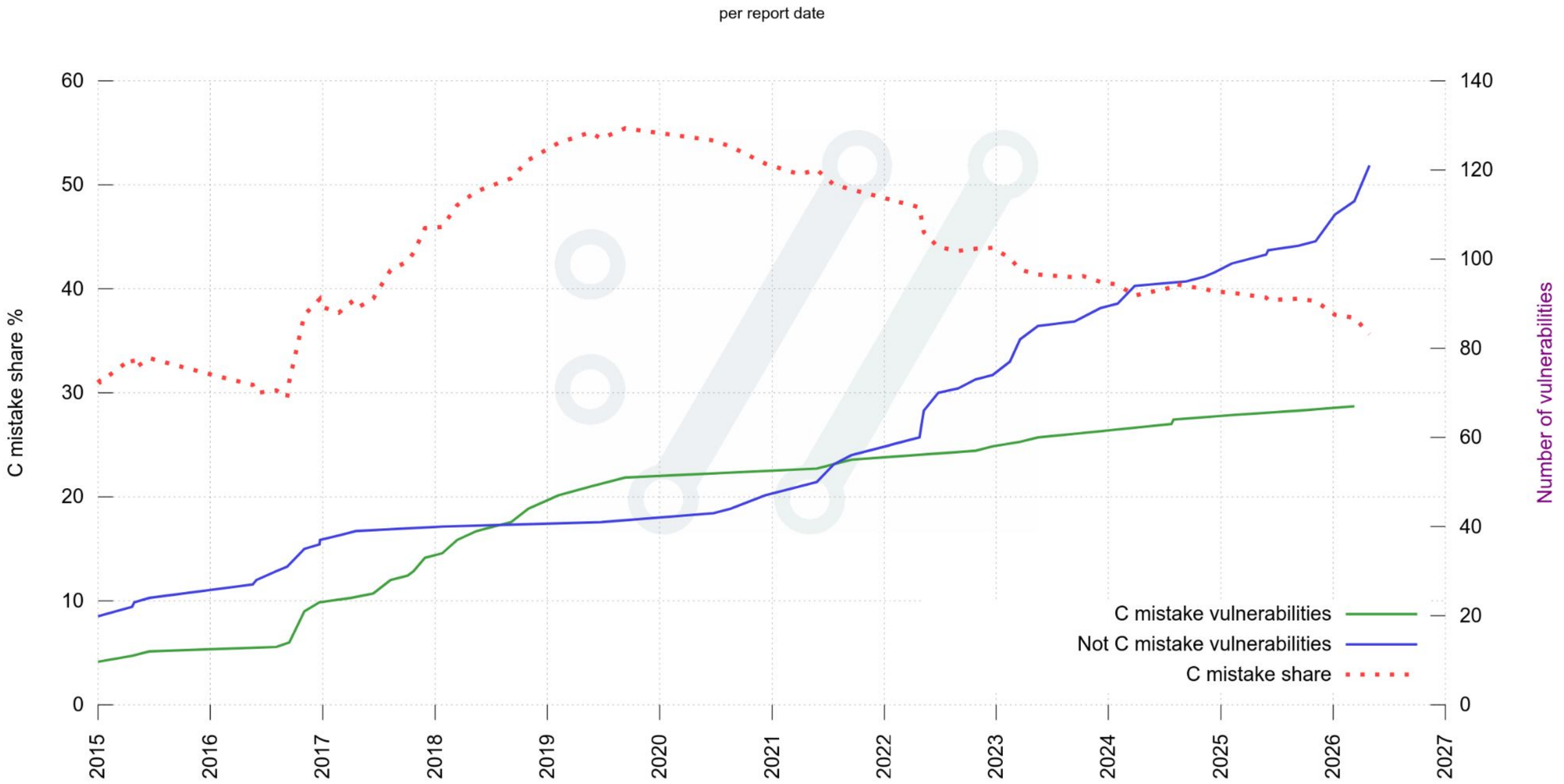
@bagder


Accumulated vulnerability age when reported
Average
Median
Number of years
0
1
2
3
4
5
6
7
8
9
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027

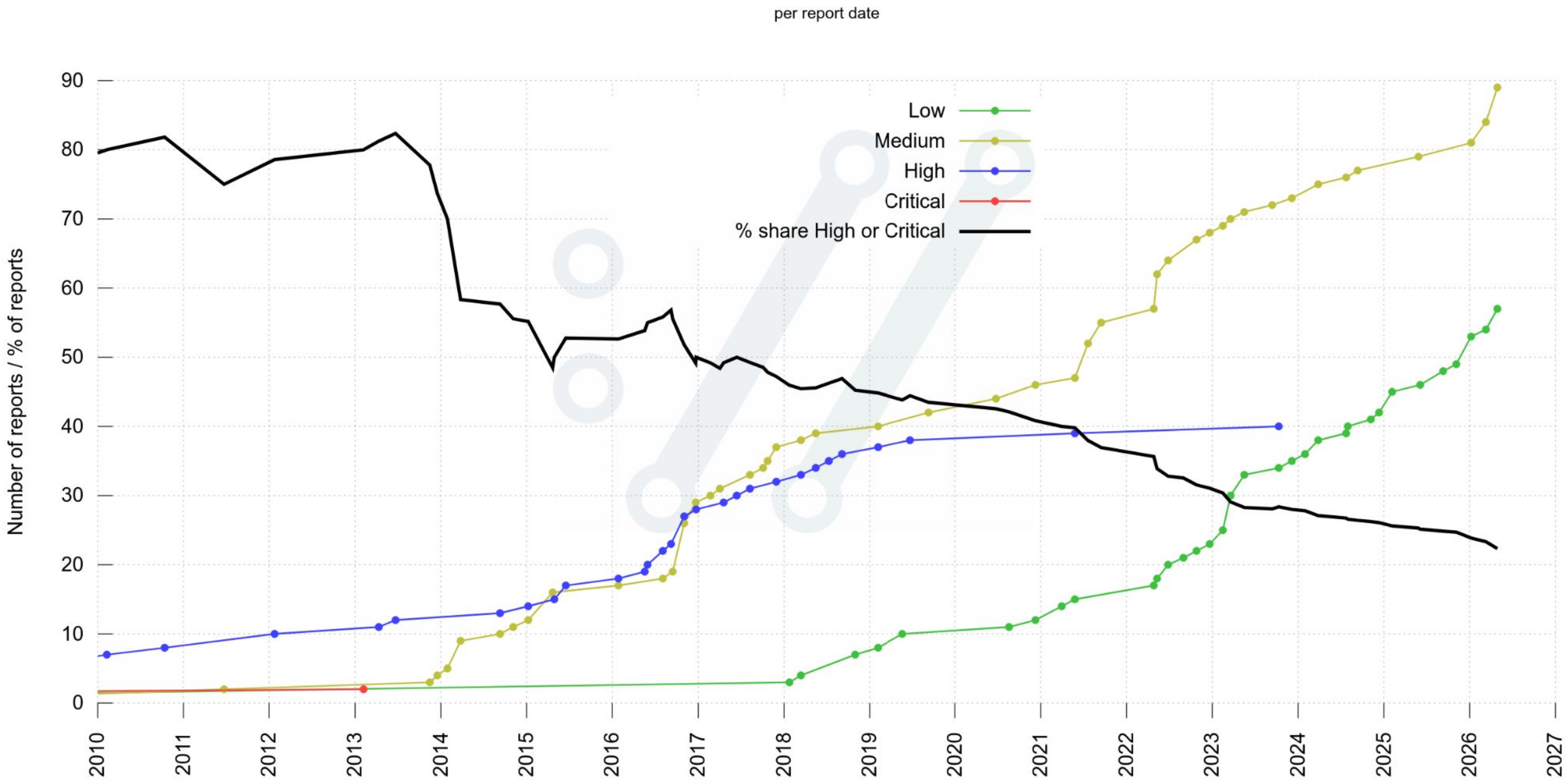
C mistakes among the vulnerabilities present in code

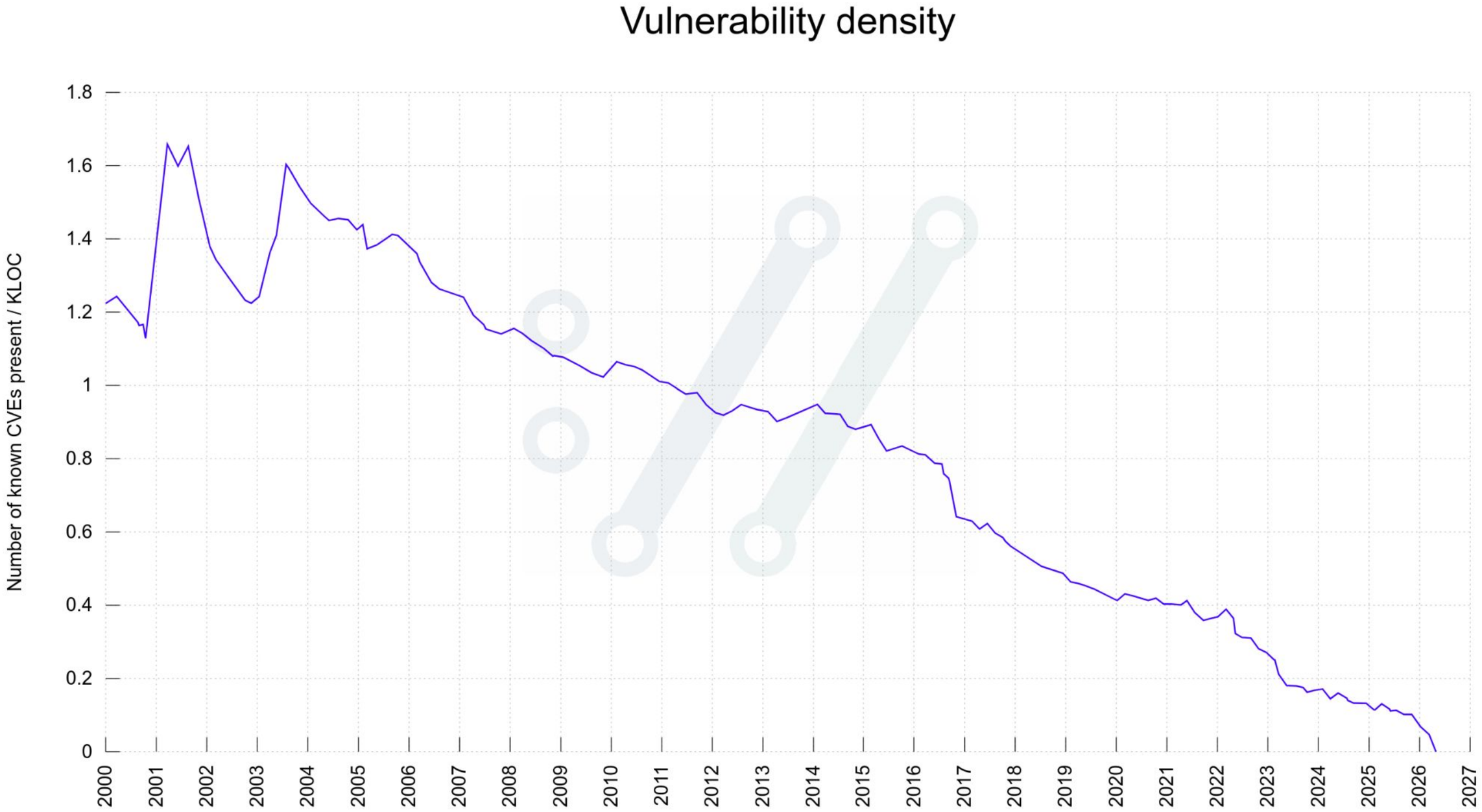


C vulnerability share

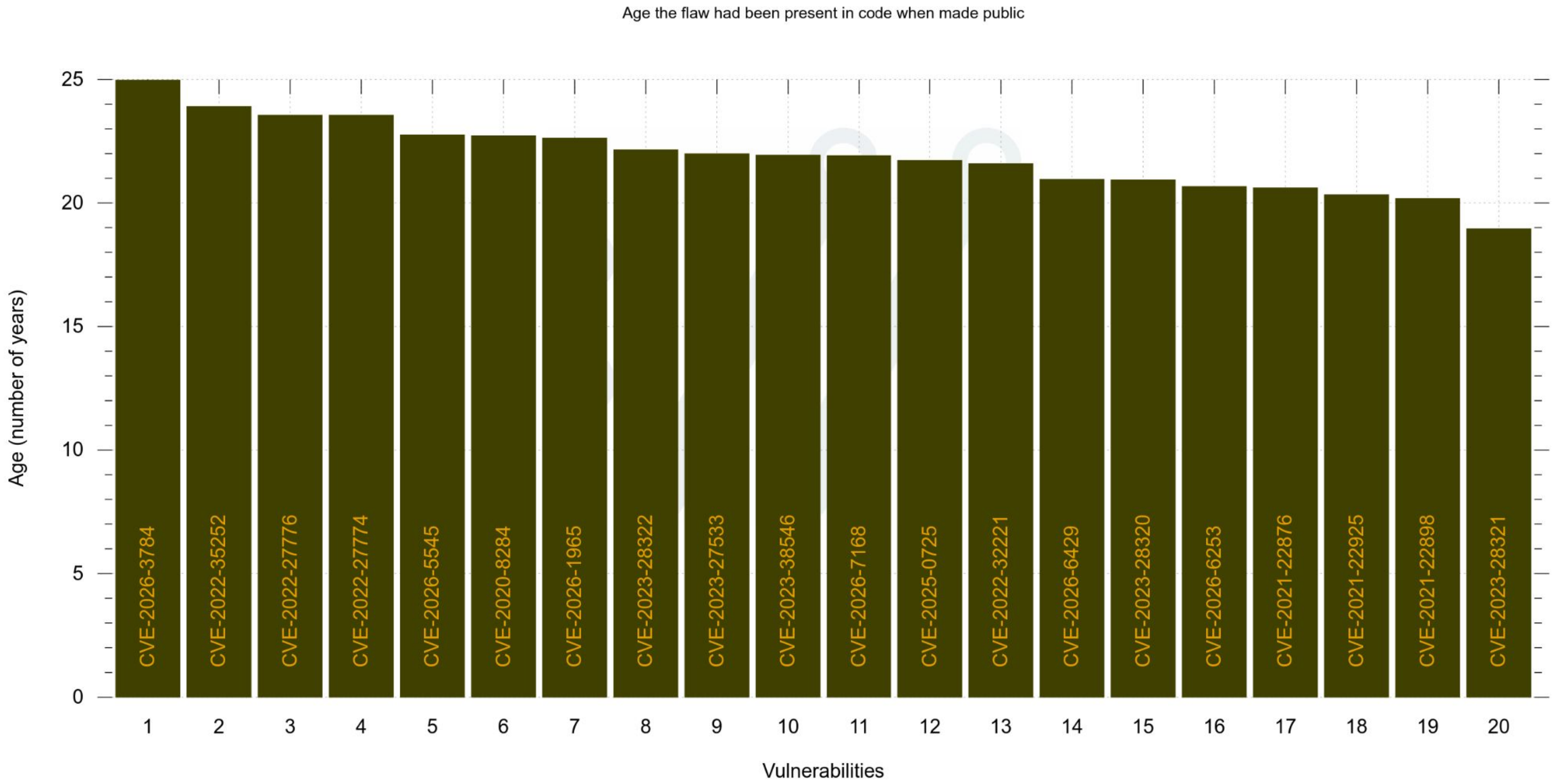


Vulnerability severity since 2010



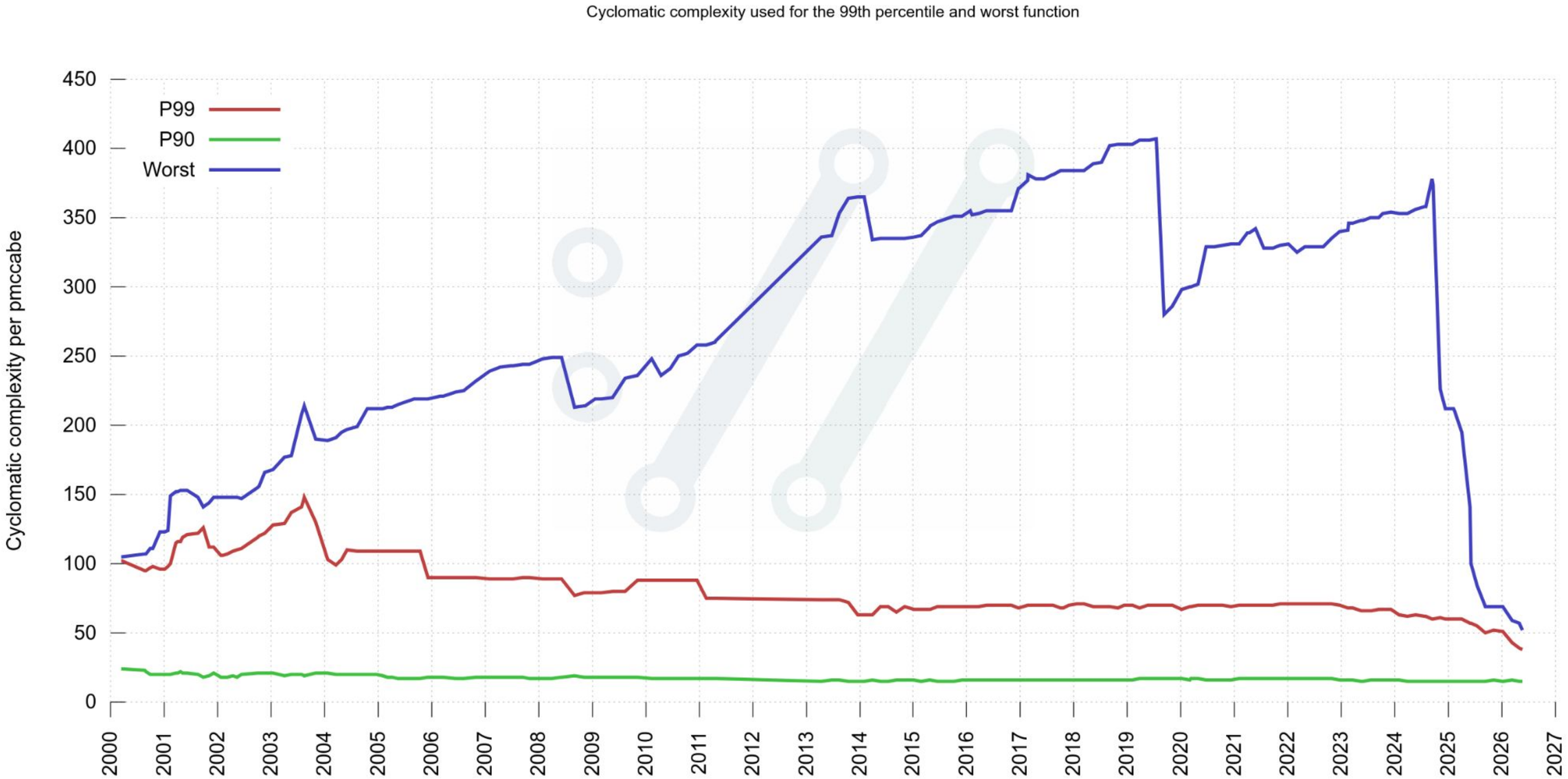

Vulnerability density
Number of known CVEs present / KLOC
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027

Top-20 longest lasting curl vulnerabilities



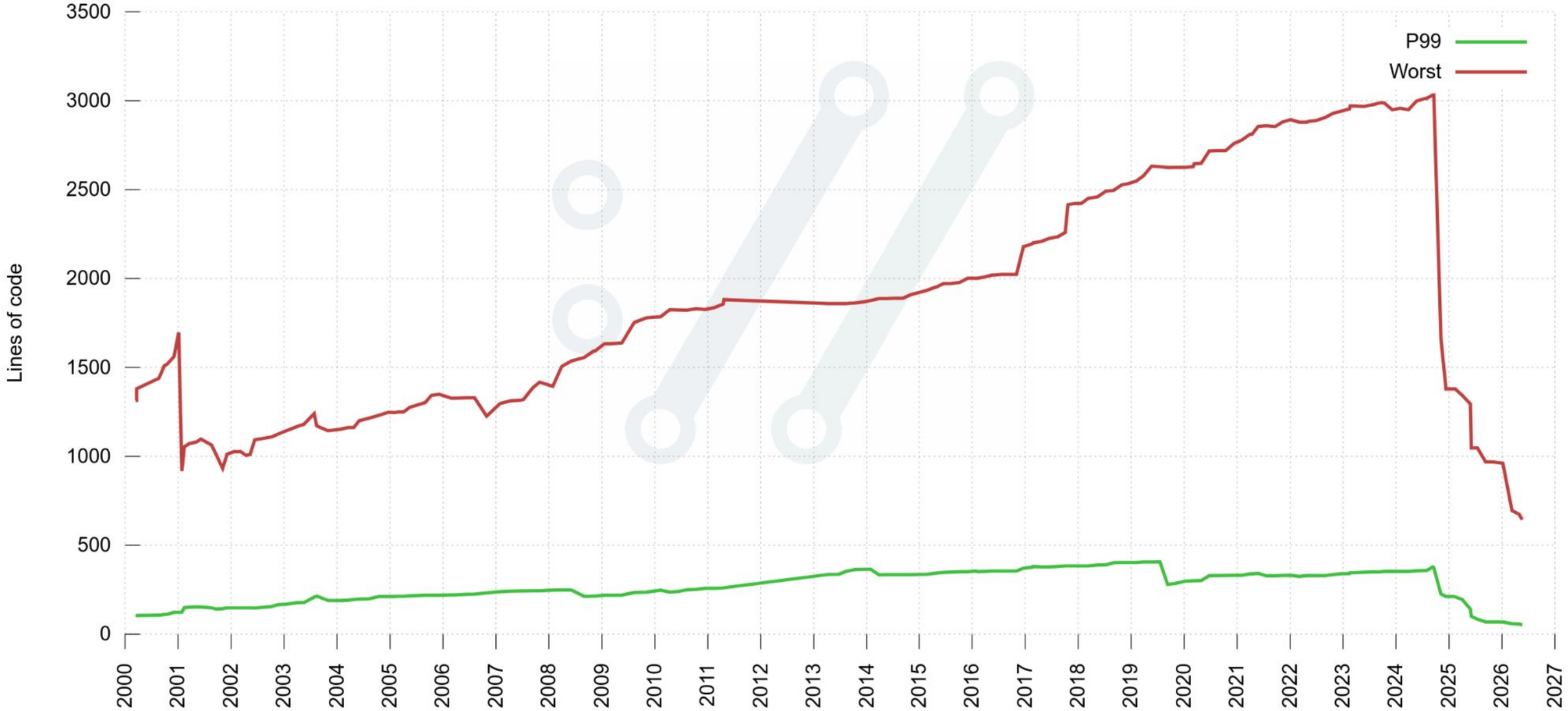
mitigations

Complexity

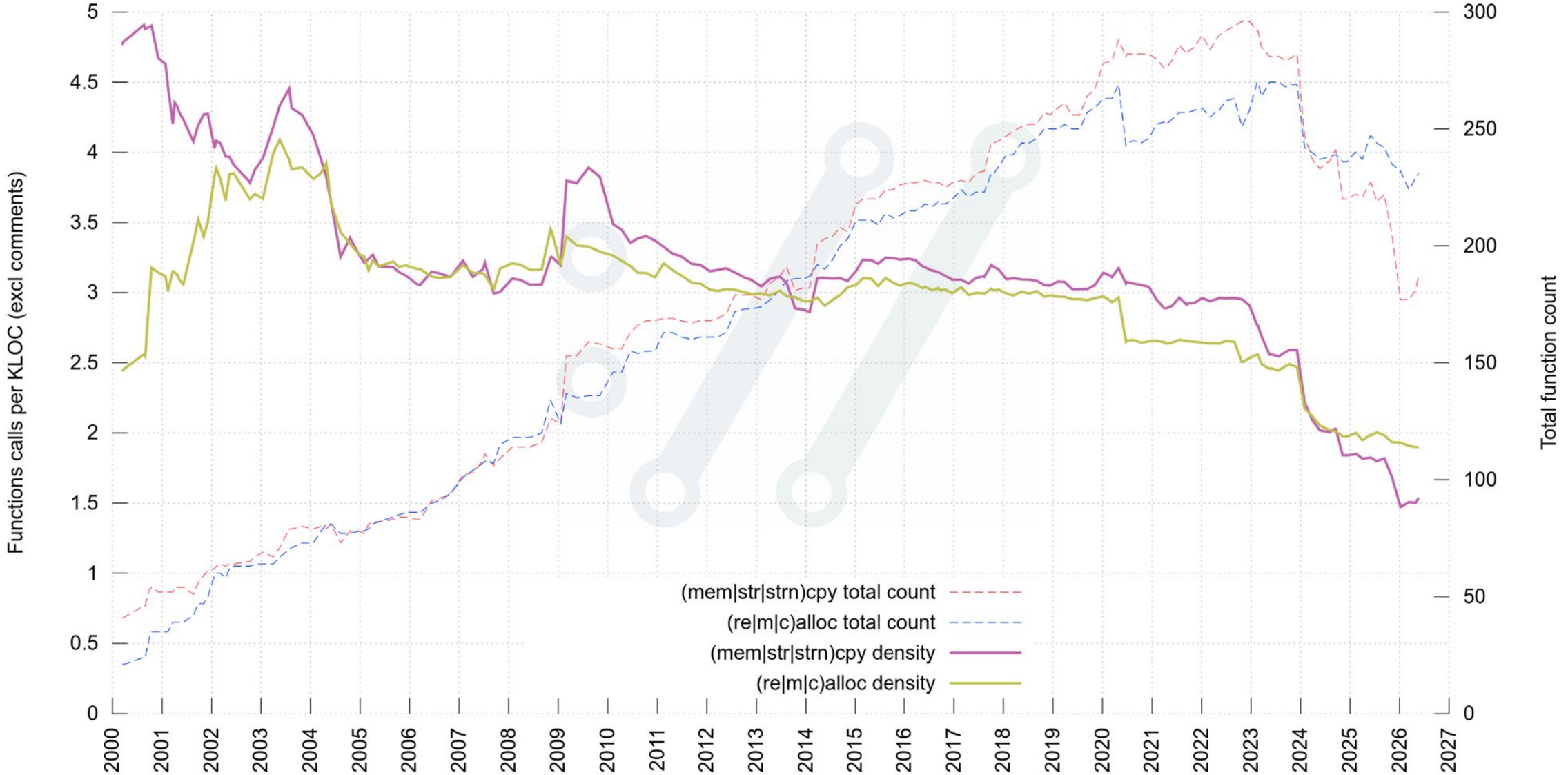

Cyclomatic complexity used for the 99th percentile and worst function
Cyclomatic complexity per pmccabe
P99
P90
Worst
450
400
350
300
250
200
150
100
50
0
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027

Function length

Lines of code for the 99th percentile and worst function. Includes comments and blank lines.



Memory function call density



What more?

~~bug bounty~~

The curl-security team

Dan Fandrich

Daniel Gustafsson

Daniel Stenberg

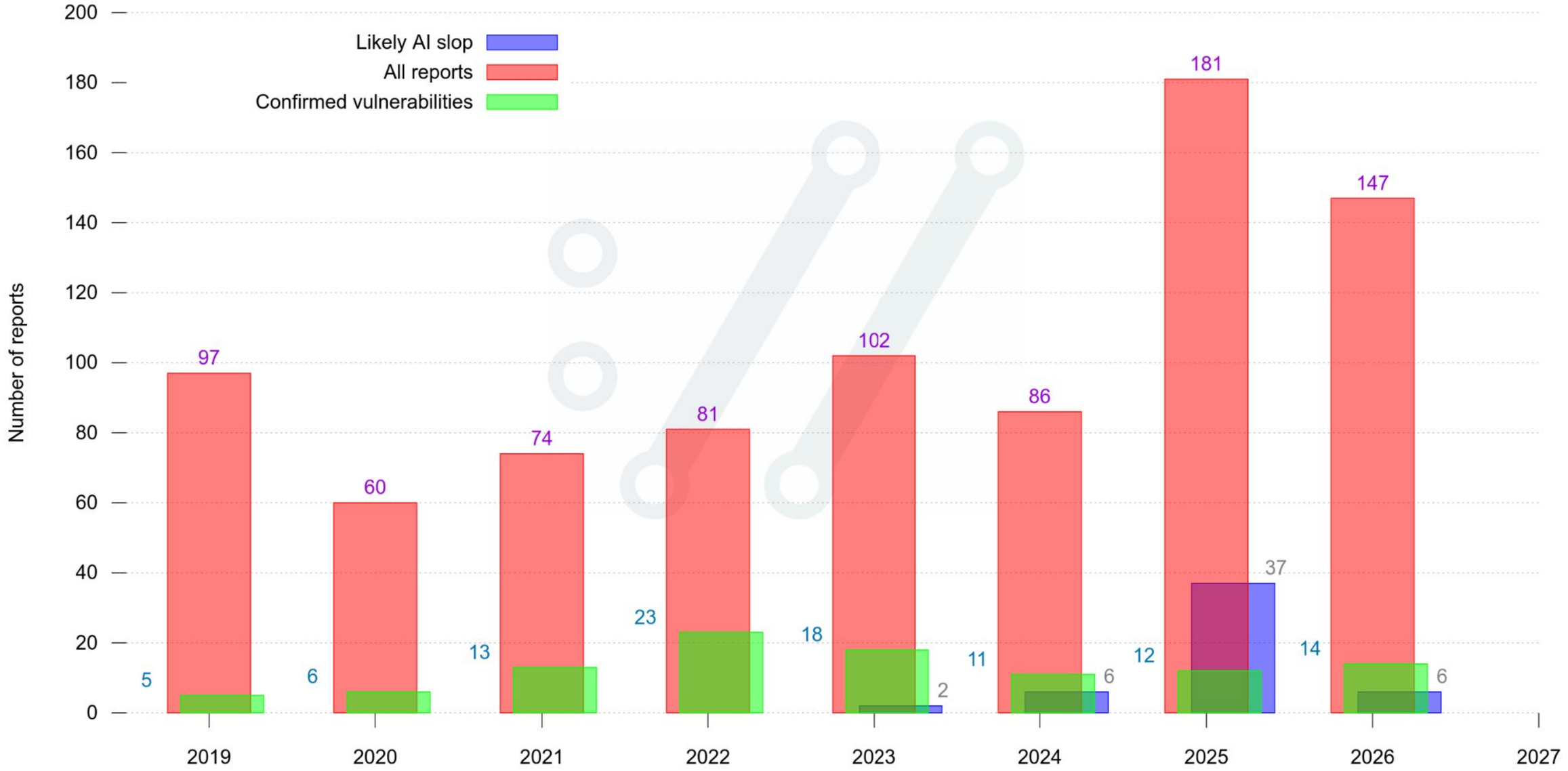
James Fuller

Max Dymond

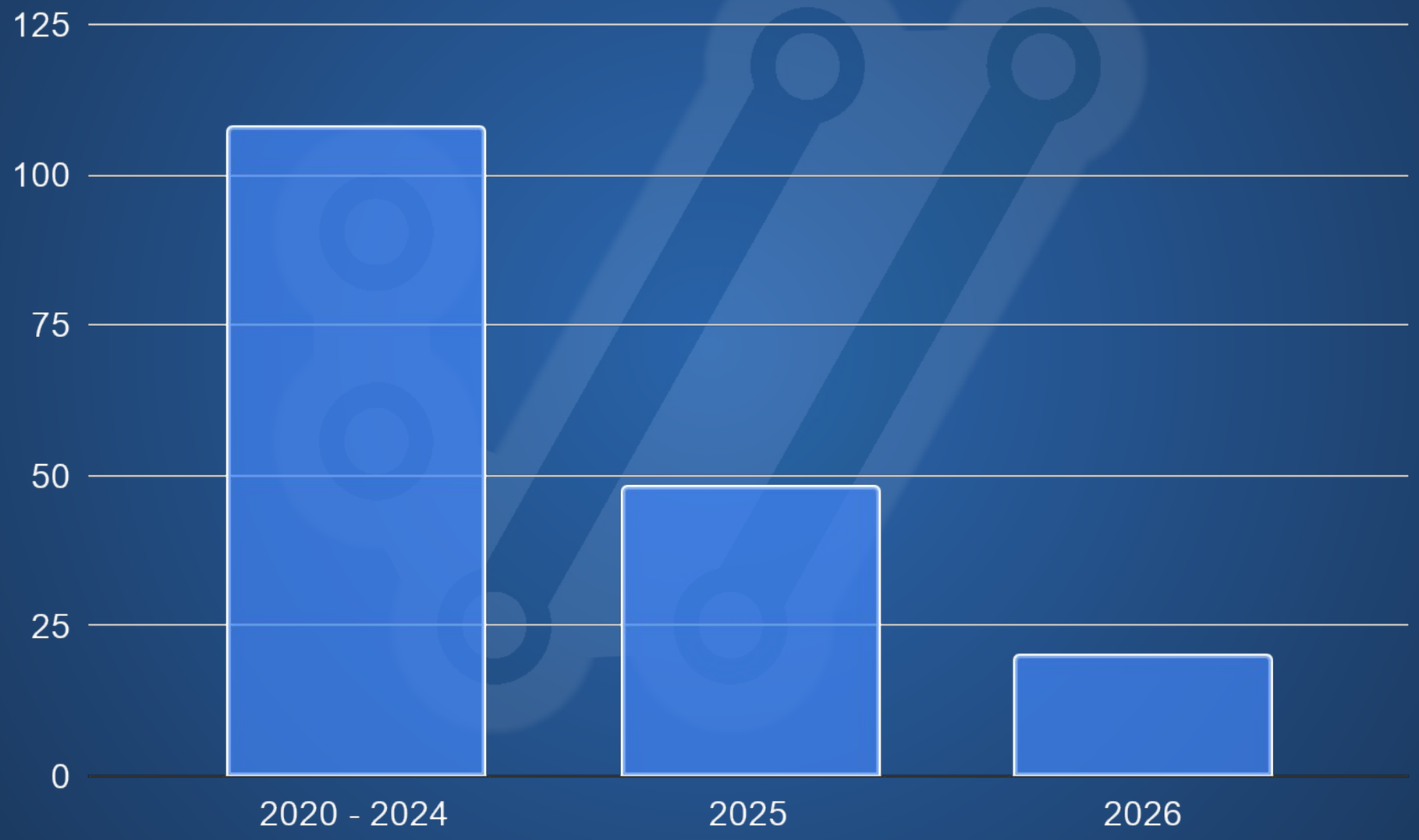
Stefan Eissing

Viktor Szakats

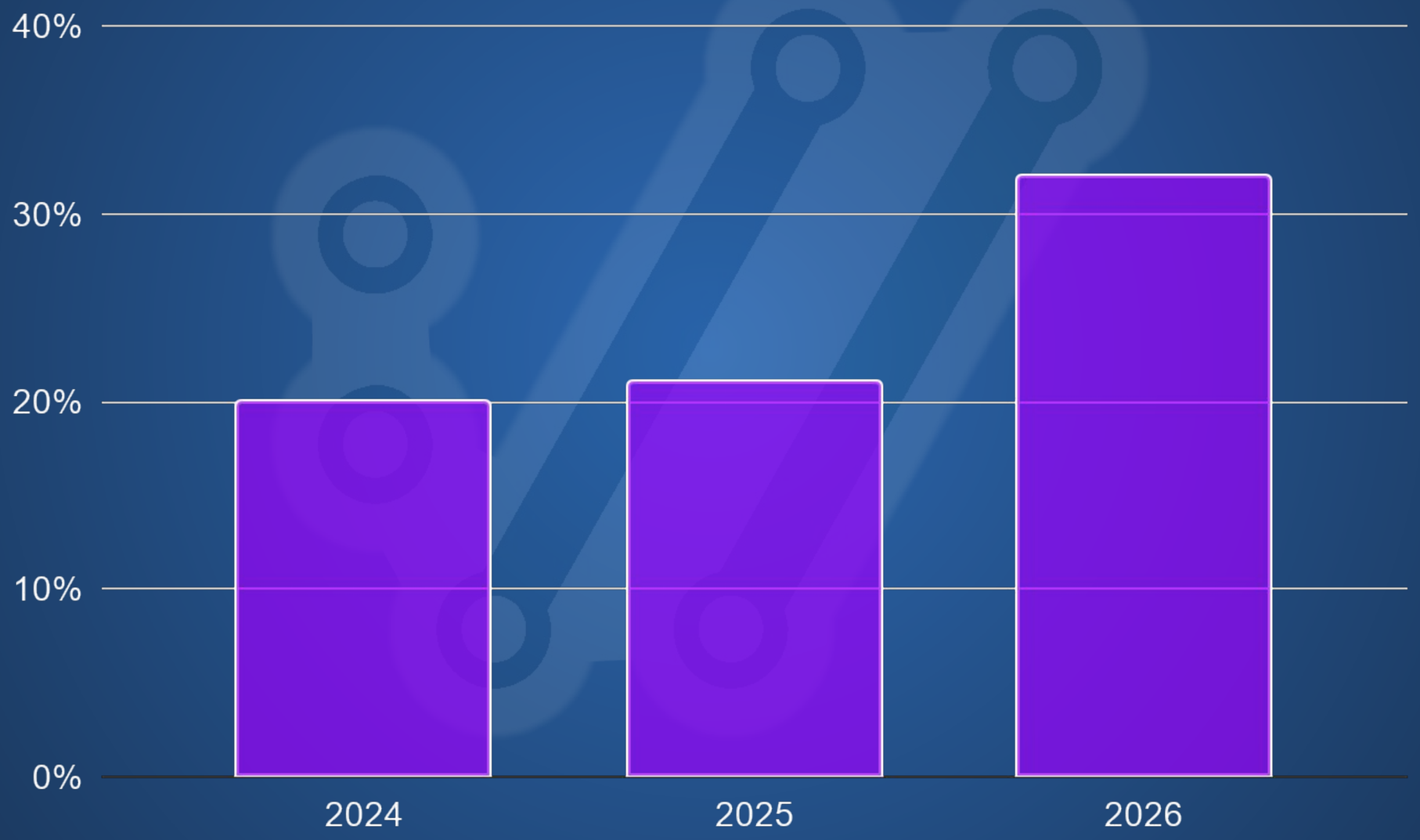
security reports on Hackerone



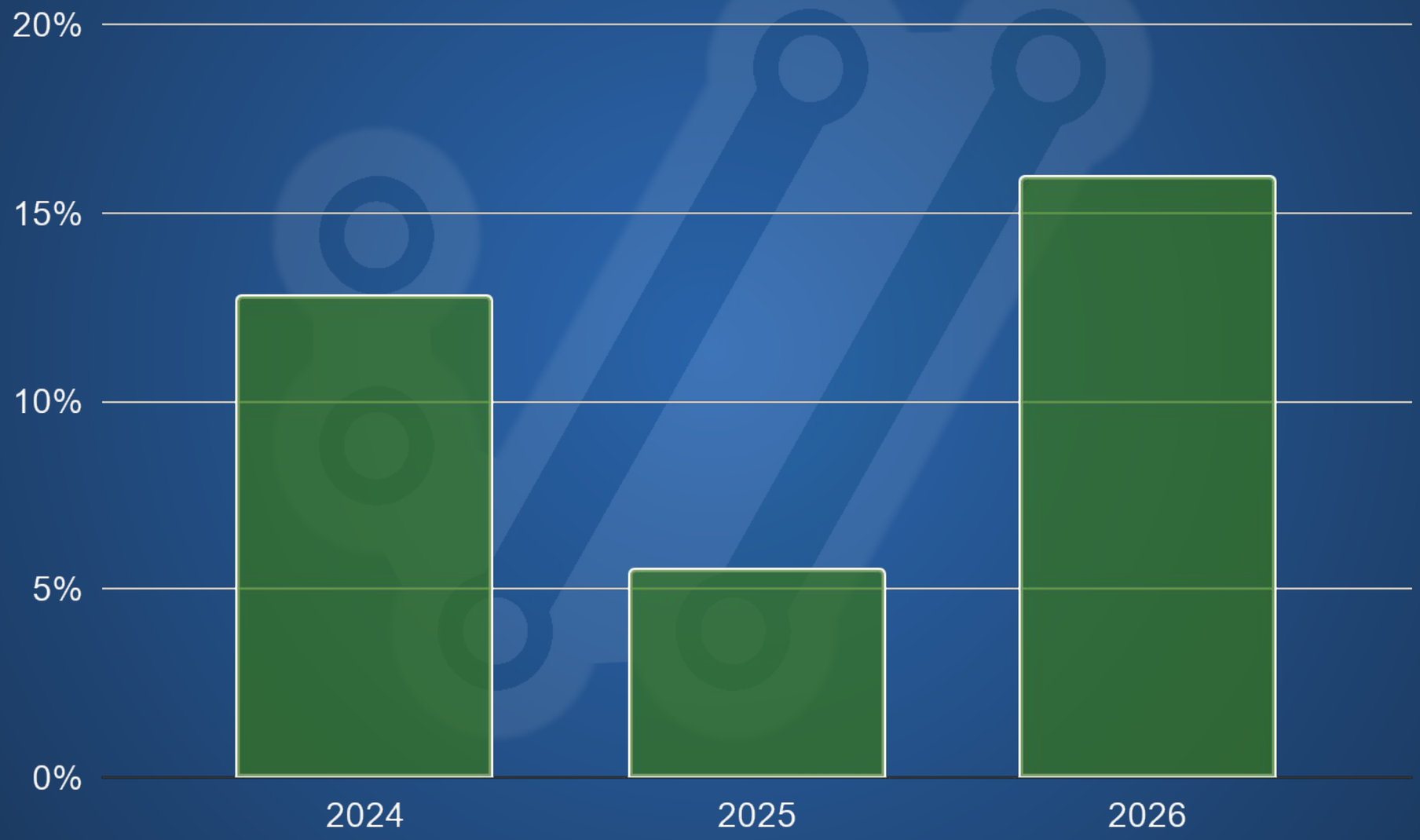
Number of hours between security reports



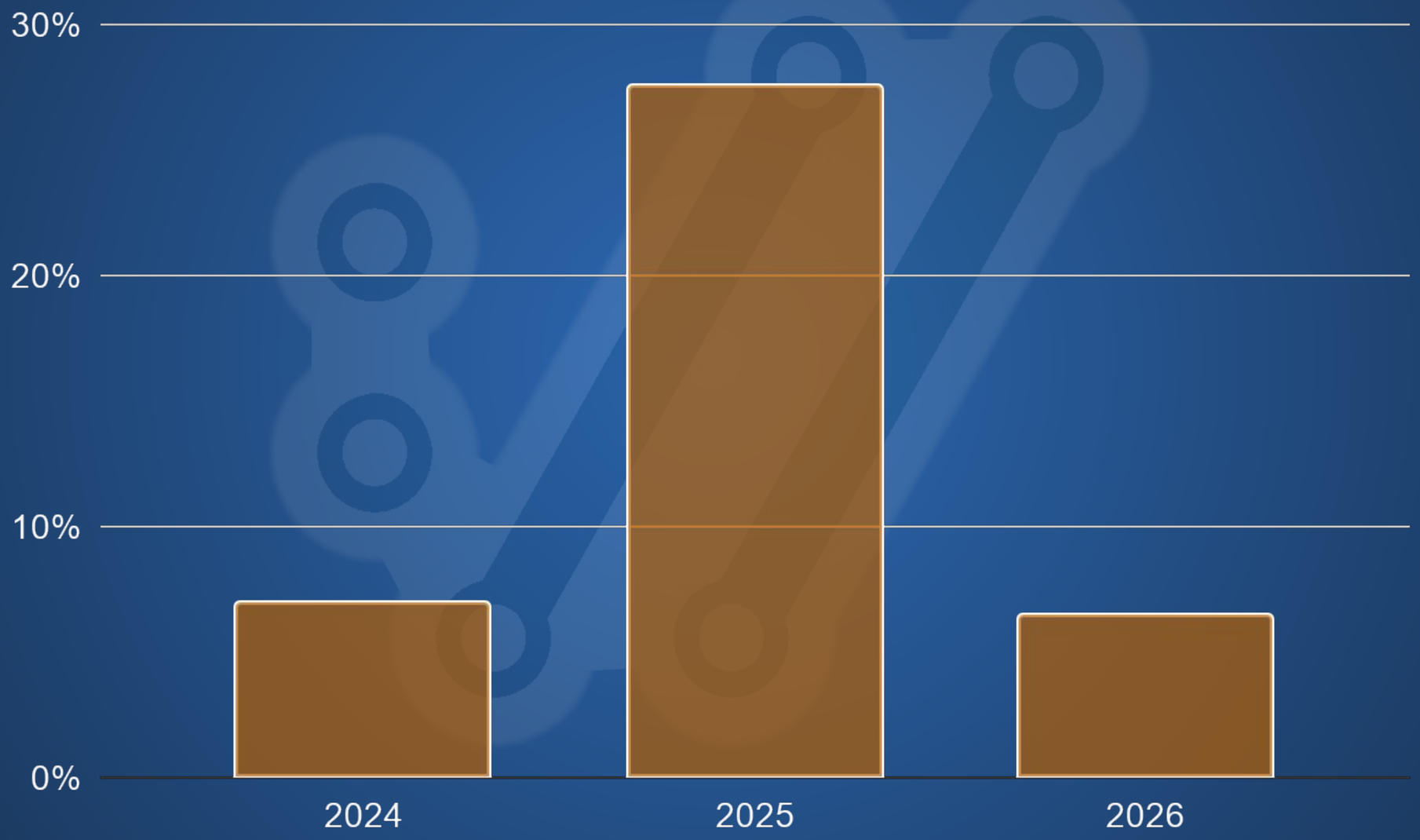
Share of reports that were bugs, not vulnerabilities



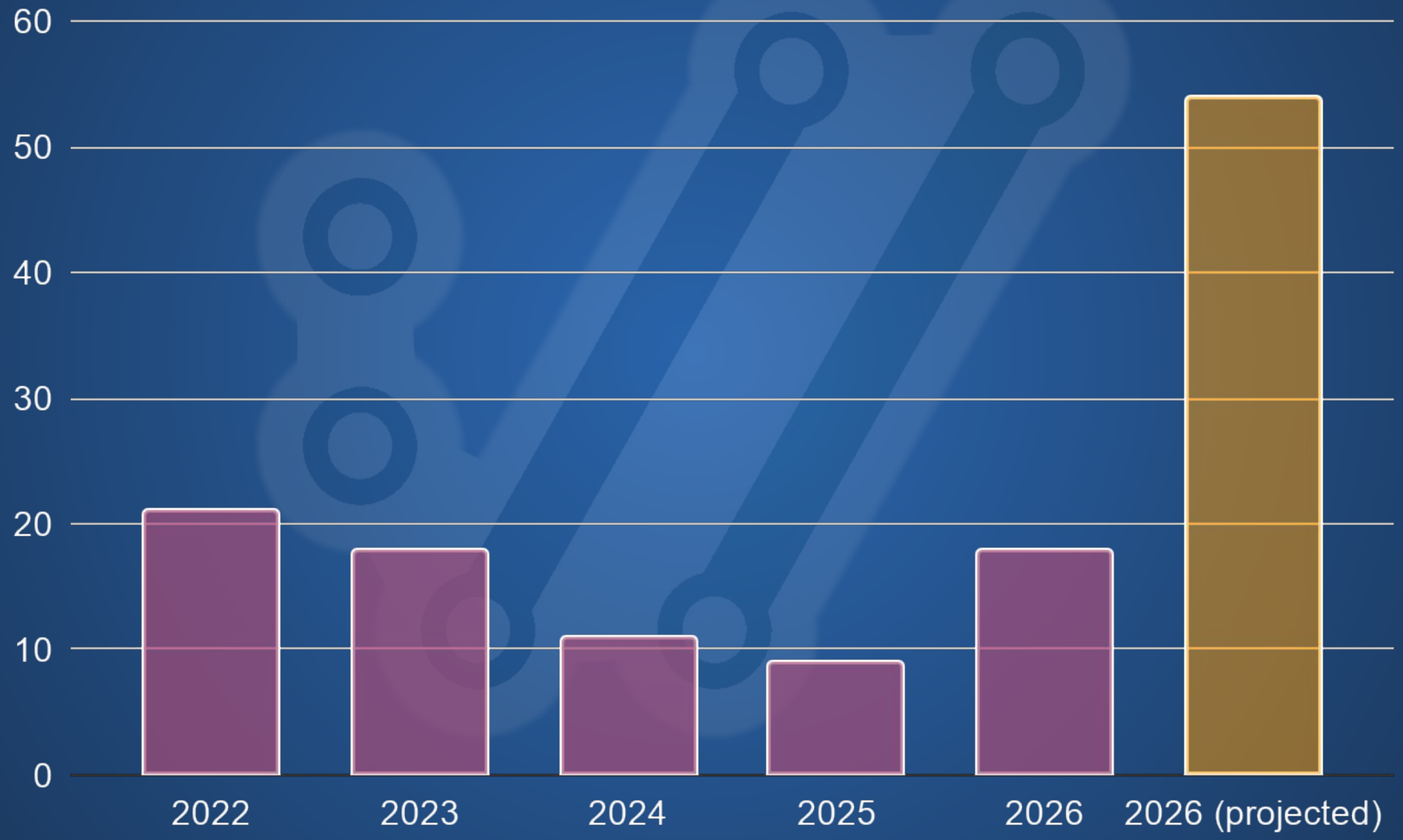
Confirmed vulnerability rate



AI slop rate



Number of published vulnerabilities



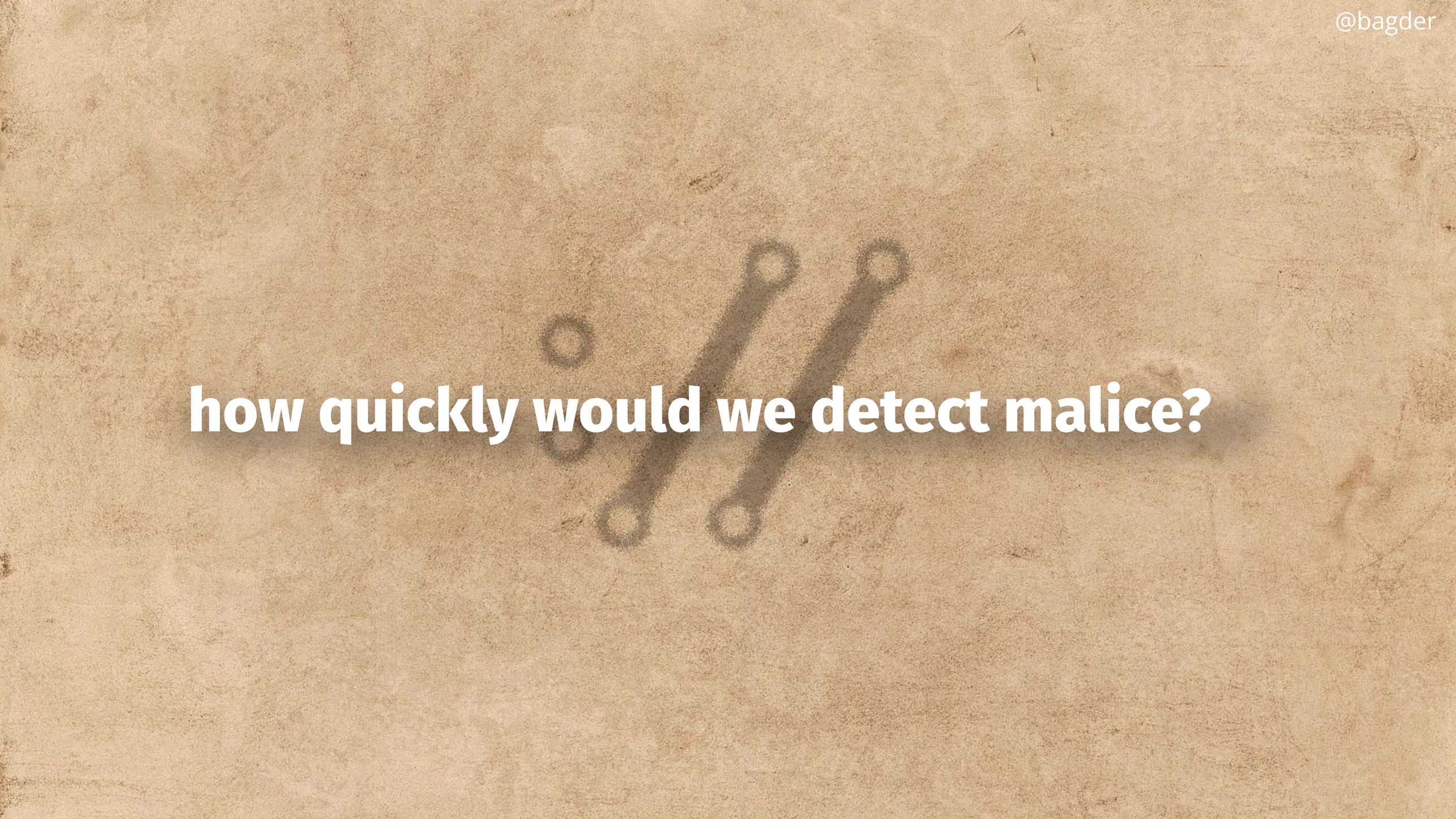
source code safety

Securing curl

Daniel Stenberg

Code style	Banned functions	Complexity checks	Human reviews	Review bots	No binary blobs	REUSE compliant
No git force push	No confusable Unicode	Document everything	Many tests	*torture tests*	CI like crazy	All picky compiler options and -Werror
Valgrind and sanitizers	AI + static code analyzers	Fuzzing, in CI and non-stop	read-only CI jobs	zizmor the CI jobs	Reproducible releases	Signed releases, commits, tags
git backup on codeberg	Vulnerabilities fixed in next release	Document vulnerabilities thoroughly	Code audits	(strong) 2fa for all committers	API and ABI stability allows *always-update*	private security reporting

Everything done in the open - accessible and transparently



how quickly would we detect malice?

server safety

Fastly secures the web front

single server point of failure

few admins

DDoS

What would a breach mean?

High-Quality Chaos


Thirty billion installations