ELISA
Enabling Linux in Safety Applications

WORKSHOP

ELISA Workshop
Munich, Germany

November 18-20, 2025
Co-hosted with Red Hat

ELISA
Enabling Linux in Safety Applications

WORKSHOP

Towards a More Sustainable and Secure Software Tooling in Free/Libre Open Source Software Environments

Dr. Stefan Tatschner, Fraunhofer AISEC
@rumpelsepp, in/stefan-tatschner

Meta

Fraunhofer AISEC

- **Stefan Tatschner (`@rumpelsepp, in/stefan-tatschner`)**
- **Security Researcher** @Fraunhofer AISEC, *Munich*, *Germany*
- **Departement**: **P**roduct Protection and **In**dustrial Security (PIN)
- **Area of studies**: software (pen)testing, code review, software supply chain
- **Partial public funding**
  - Institute level: industry 33% vs. research 66% (source: annual report 2023)
- **Collaboration with OEMs and TIER-1 suppliers**
- **Thesis: https://doi.org/10.34961/18737**

https://sdgs.un.org/

https://www.ul.ie/media/25857/download?inline=

Mission-Driven Learning

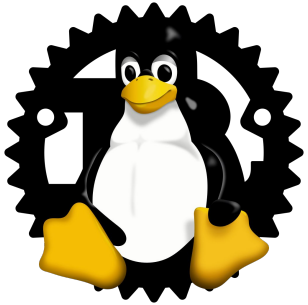
By 2030, UL will have pioneered mission-driven curriculum to support the transition

Transitioning to a sustainable society will require a new generation of change-makers. This mission sees UL ensure that relevant sustainability-led theory and associated practices are included in all curriculum. In doing so, each learner that studies at UL will graduate with a sustainability mindset and the capacity to contribute to a more sustainable world.

Sustainability

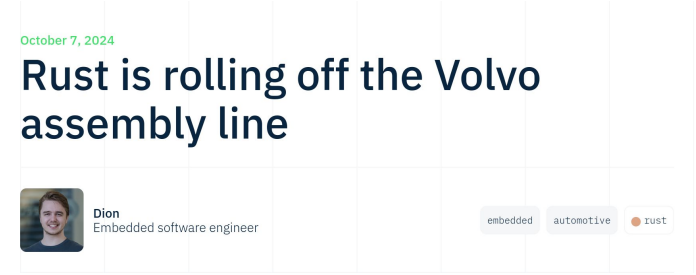
Software Architecture Sustainability: Ability to endure by evolving (H. Koziolek, 2011. DOI: 10.1145/2000259.2000263)

Key Challenge: Universally solve problems at a lower layer

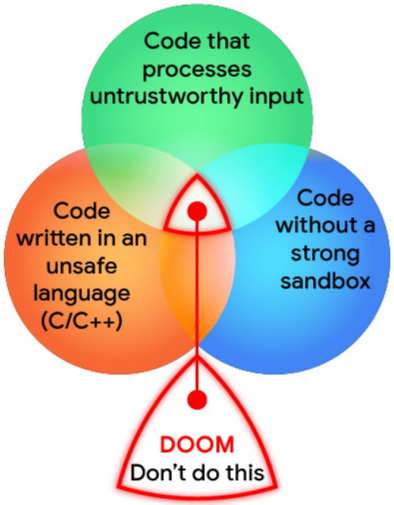


CC0

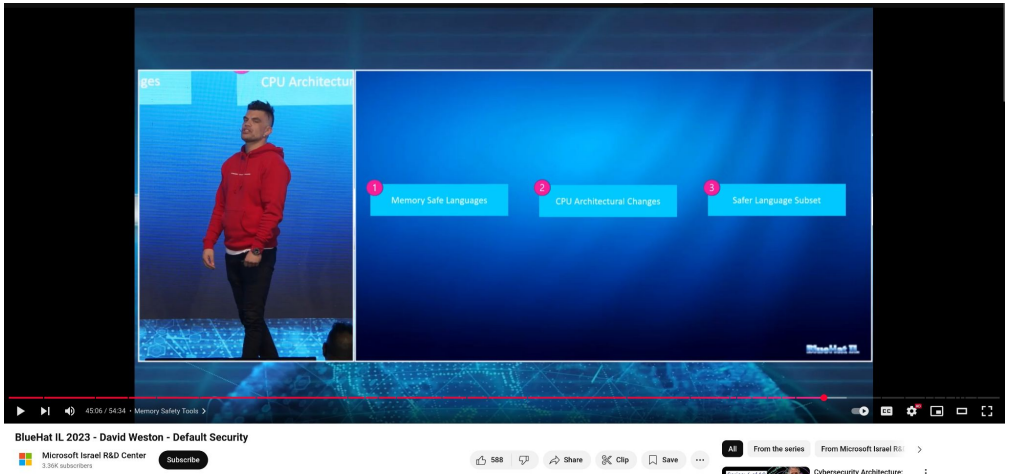
https://rust-for-linux.com/



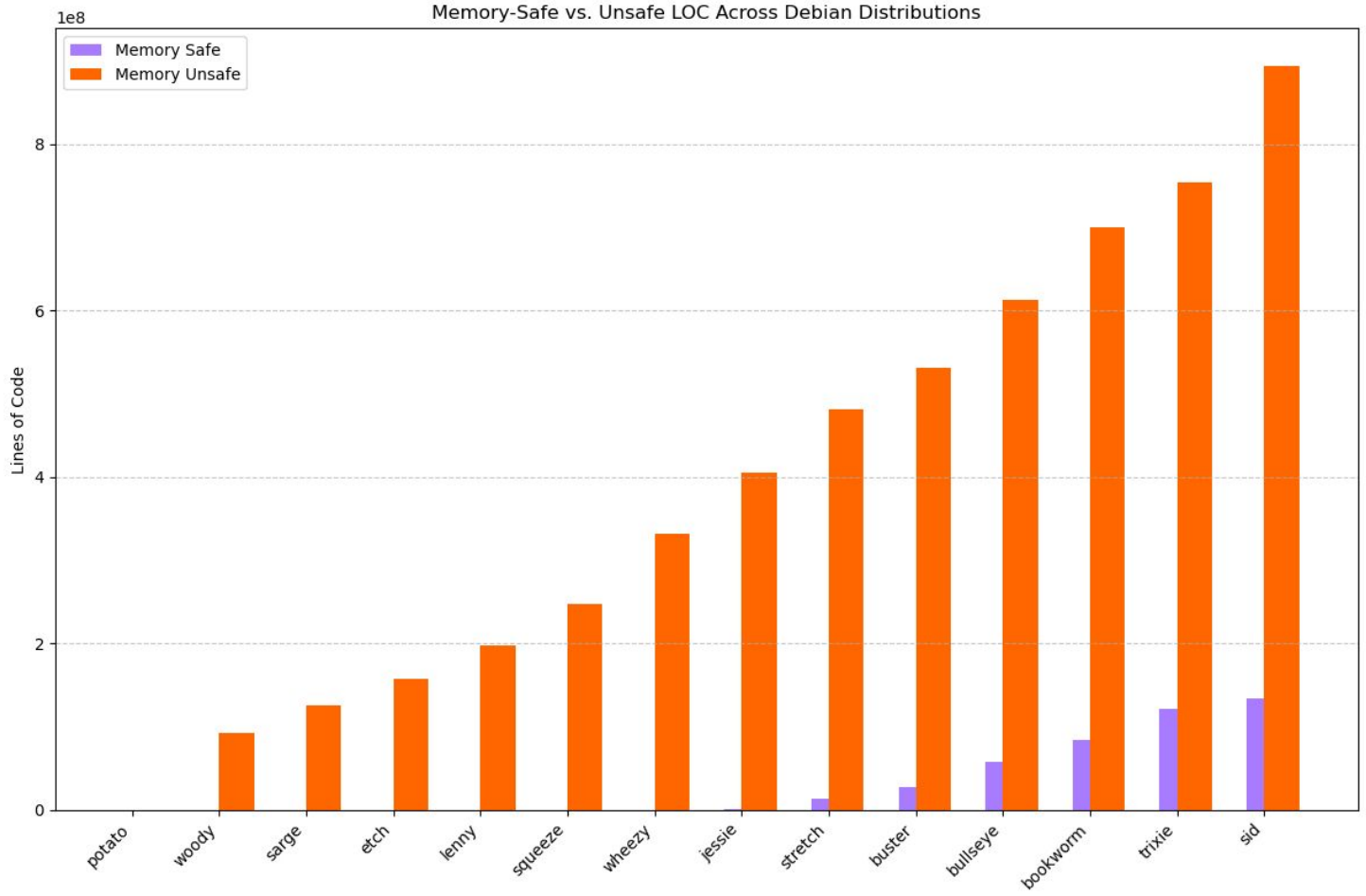
https://tweedegolf.nl/en/blog/137/rust-is-rolling-off-the-volvo-assembly-line

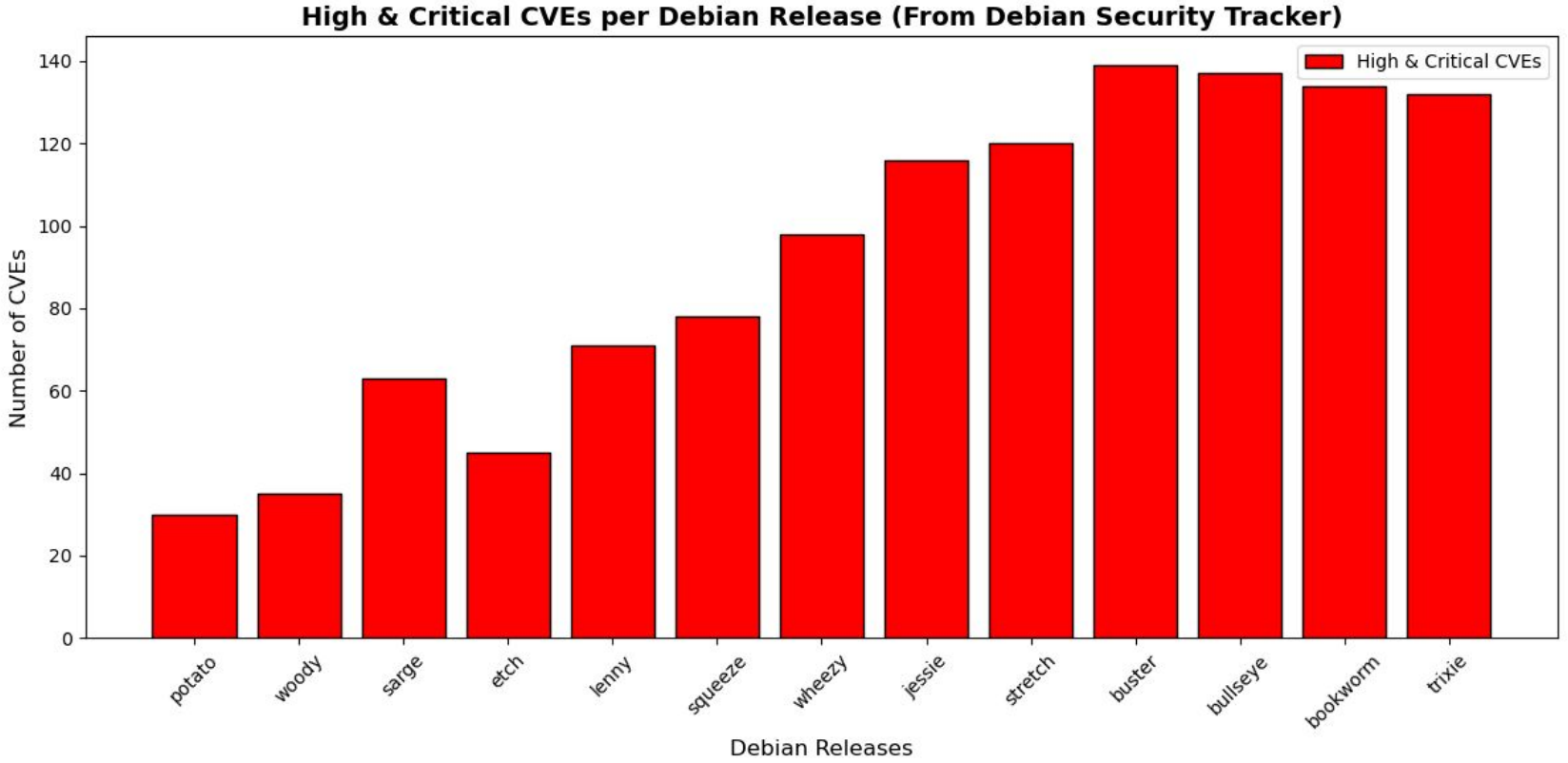


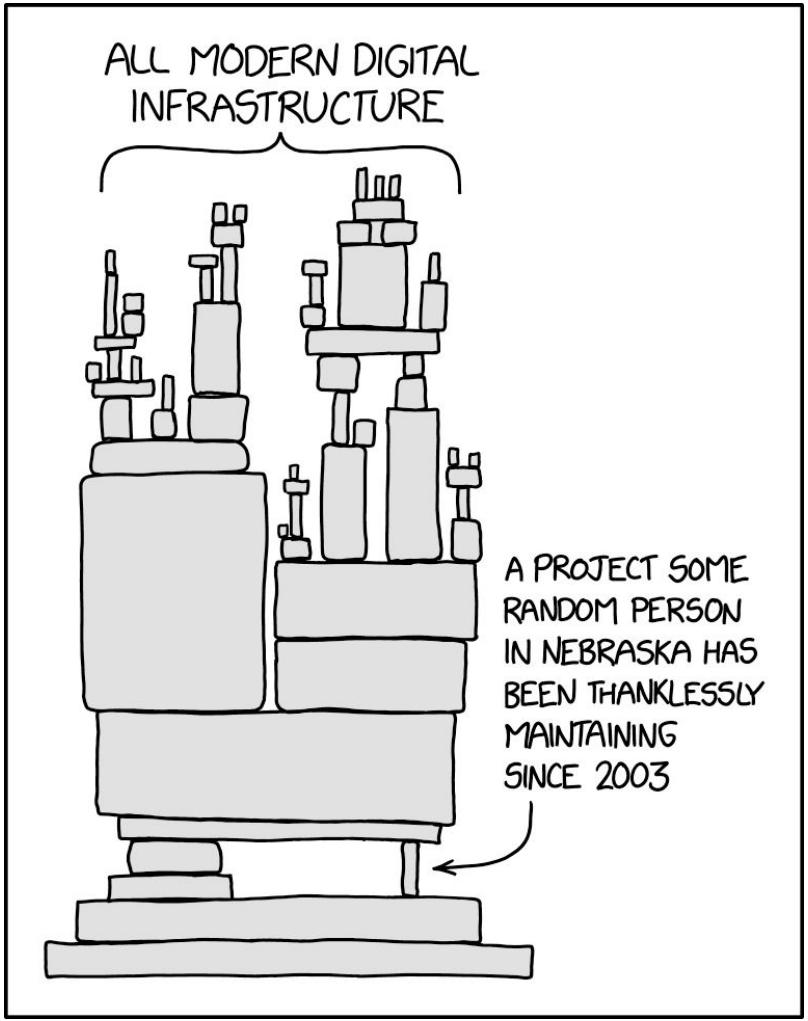
https://security.googleblog.com/2021/04/rust-in-android-platform.html



https://www.youtube.com/watch?v=8T6ClX-y2AE


Memory-Safe vs. Unsafe LOC Across Debian Distributions
1e8
Memory Safe
Memory Unsafe
Lines of Code
0
2
4
6
8
potato
woody
sarge
etch
lenny
squeeze
wheezy
jessie
stretch
buster
bullseye
bookworm
trixie
sid


High & Critical CVEs per Debian Release (From Debian Security Tracker)
High & Critical CVEs
Number of CVEs
0
20
40
60
80
100
120
140
potato
woody
sarge
etch
lenny
squeeze
wheezy
jessie
stretch
buster
bullseye
bookworm
trixie
Debian Releases



XZ Utils backdoor

Lasse Collin

CVE-2024-3094

Shellshock

CC BY 3.0

CVE-2014-6271

Log4Shell

Bundesamt für Sicherheit in der Informationstechnik

Nationales IT-Lagezentrum BSI

SCHWACHSTELLE | GEFÄHRDUNG | VORFALL | IT-ASSETS

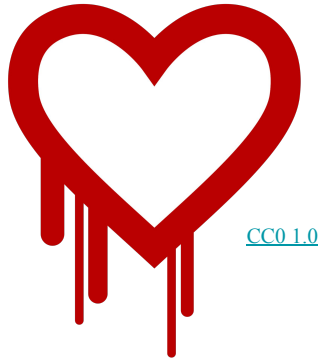
Kritische "Log4Shell" Schwachstelle in weit verbreiteter Protokollierungsbibliothek Log4j (CVE-2021-44228)

CSW-Nr. 2021-549177-1232, Version 1.2, 12.01.2022

IT-Bedrohungslage*: 2 / Gelb

CVE-2021-44228

Heartbleed



CC0 1.0

CVE-2014-0160

Home › Notes › VU#952657

Rsync contains six vulnerabilities

Vulnerability Note VU#952657

Original Release Date: 2025-01-14 | Last Revised: 2025-03-13

Vulnerability Note VU#952657

What are the current challenges in the FLOSS domain?

A Quic(k) Security Overview: A Literature Research on Implemented Security Recommendations

RQ: “Do complex standard documents and numerous different implementations contribute to insecure software ecosystems?”

https://doi.org/10.1145/3600160.3605164

Project	★	Backed by	Language	TLS Stack	References
quic-go	8.0k	Google	Go	modified Go stdlib	Syncthing, caddy, cloudflared, traefik
cloudflare/quiche	7.4k	Cloudflare	Rust	Boring SSL	Android (DNS), curl, nginx
MsQuic	3.3k	Microsoft	C	SChannel	HTTP/3 and SMB Stack on Windows
Quinn	2.7k	Instant Domains, Inc	Rust	multiple (default: rustls)	hyper/reqwest
Neqo	1.6k	Mozilla	C, C++	NSS	Firefox
XQUIC	1.4k	Alibaba	C	BoringSSL or BabaSSL	tengine
mvfst	1.3k	Facebook	C++	Fizz	internally at Facebook
aioquic	1.2k	Jeremy Lainé	Python	custom	hypercorn, httpx
LSQUIC	1.2k	LiteSpeed Technologies Inc	C	BoringSSL	internally at LiteSpeed
ngtcp2	0.9k	Tatsuhiro Tsujikawa	C++	multiple	curl
s2n-quic	0.9k	Amazon Web Services	Rust	s2n-tls or rustls	Amazon Web Services
quicly	0.6k	Fastly, DeNA Co. Ltd.	C	BoringSSL	H2O webserver
google/quiche	0.4k	Google	C, C++	BoringSSL	Chromium

Security Consideration	quic-go	[illegible]	[illegible]	[illegible]	[illegible]	XQUIC*	mvfst*	aioquic	LSQUIC	ngtcp2	s2n-quic	quicly	g/quiche*
Amplification (cf. Paragraph 3.1.1.2)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Optimistic ACK (cf. Paragraph 3.1.1.2)	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗
Slowloris (cf. Paragraph 3.1.1.2)	✓	✗	✗	✗					✓	✗	✓	✗	✗
Stream Fragmentation and Reassembly (cf. Paragraph 3.1.1.2)	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Stream Commitment (cf. Paragraph 3.1.1.2)	✓	✓	✓	✓	✓	✓	✓				✓	✓	✓
Peer Denial of Service (cf. Paragraph 3.1.1.2)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	[✓]	✗	✗
Explicit Congestion Notification (cf. Paragraph 3.1.1.2)	n/a	n/a	✓	✓	n/a	n/a	n/a	n/a	n/a	✓	✓	n/a	✓
Stateless Reset Oracle (cf. Paragraph 3.1.1.2)	✗							✗	✗	✗	✗	✗	✗
Version Downgrade (cf. Paragraph 3.1.1.2)	✓	✗	✓	✗	✓	✗							
Traffic Analysis (cf. Paragraph 3.1.1.2)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Development in private…

Complex topic…

Complex topic…

Required feature not implemented…

Required by RFC 9250 (DNS over QUIC)

Implementation	Amount of Malicious Connections Opened	Container Health	Responsiveness to Benign Clients
aioquic	33	✗	✗
LSQUIC	10000	✓	✓
MsQuic	10000	✓	✓
mvfst	85	✗	✗
Neqo	762	✗	✗
nginx	10000	✓	✓
ngtcp2	10000	✓	✓
Picoquic	256	✓	✗
quic-go	10000	✓	✓
quiche	33	✗	✗
quicly	6547	✓	✓
Quinn	33	✗	✗

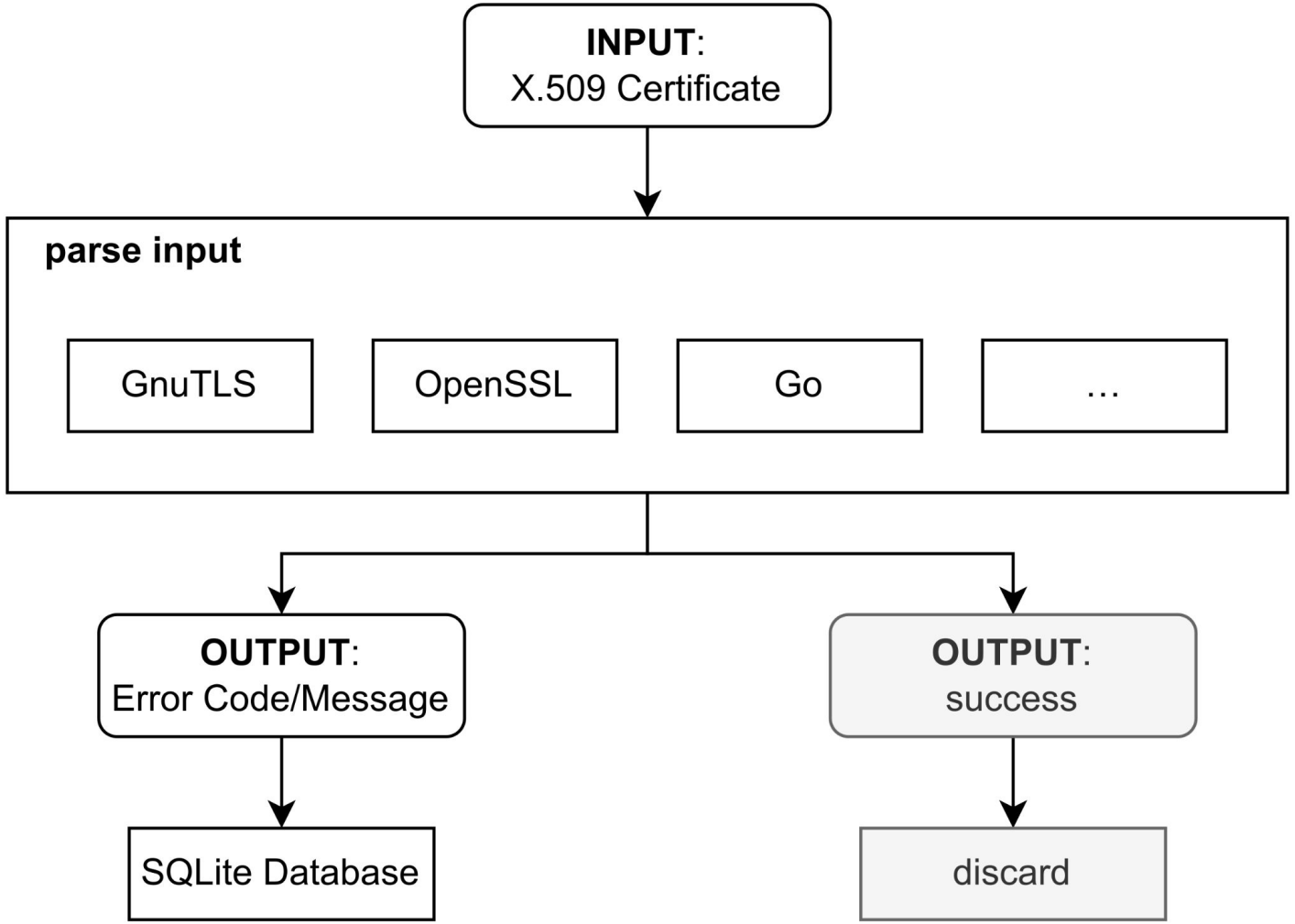
✓ Server appears to implements effective countermeasures.
✗ Server does not appear to implement any countermeasures.

Kacper Darowski, Ba-Thesis, *Not Quite As QUICk: Automatic Vulnerability and RFC Security Considerations Compliance Testing of QUIC Protocol Server Implementations*

ParsEval: Evaluation of Parsing Behavior using Real-world Out-in-the-wild X.509 Certificates

RQ: “Do complex standard documents and numerous different implementations contribute to insecure software ecosystems?”

https://doi.org/10.1145/3664476.3669935


INPUT:
X.509 Certificate
parse input
GnuTLS
OpenSSL
Go
…
OUTPUT:
Error Code/Message
OUTPUT:
success
SQLite Database
discard

N = 186,576,846

Library	n_e	$r_e = \frac{n_e}{N}$
wolfSSL	24,196,189	13.16%
Mbed TLS	24,562,224	12.97%
Go stdlib	51,181	0.03%
GnuTLS	28,875	0.02%
cryptography	21,492	0.01%
OpenSSL	4,803	0.003%

Lang.	Library	Version	*ASN1_PARSE_ERROR*	*CRYPTO_UNSUPPORTED*	*CRYPTO_VALUE_ERROR*	*UNCATEGORIZED*	*X509_PARSE_ERROR*	*X509_UNSUPPORTED*	*X509_VALUE_ERROR*	***Sum***
C	GnuTLS	3.8.1	6,321	n.a.	n.a.	n.a.	12,599	n.a.	9,955	**28,875**
		3.8.3	6,321	n.a.	n.a.	n.a.	12,599	n.a.	9,955	**28,875**
	Mbed TLS	3.4.1	84,397	183	1,561	n.a.	17,192	8,553	24,085,344	**24,197,230**
		3.5.2	84,411	183	1,561	n.a.	17,227	8,553	24,092,409	**24,204,344**
	OpenSSL	3.0.10	4,803	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	**4,803**
		3.0.13	4,803	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	**4,803**
	wolfSSL	5.5.4	114,291	23	n.a.	11,426	n.a.	n.a.	24,070,449	**24,196,189**
		5.7.0	467,257	23	n.a.	41	n.a.	n.a.	24,094,903	**24,562,224**
Go	stdlib	1.20.7	40,736	46	374	n.a.	5,148	n.a.	4,877	**51,181**
		1.22.1	40,736	46	374	n.a.	5,148	n.a.	4,877	**51,181**
Python	cryptography	41.0.4	19,982	n.a.	n.a.	n.a.	n.a.	n.a.	1,510	**21,492**
v3.11		42.0.5	19,982	n.a.	n.a.	n.a.	n.a.	n.a.	1,510	**21,492**

(a) Number of unique certificates that were classified with “X509: invalid version”.

Library	Version	Count
GnuTLS	3.8.1	0
Mbed TLS	3.5.2	160
OpenSSL	3.0.13	0
wolfSSL	5.7.0	160
Go stdlib	1.22.1	160
cryptography	42.0.5	160

(b) Number of unique certificates that were classified with “x509: invalid RSA public exponent”.

Library	Version	Count
GnuTLS	3.8.1	3
Mbed TLS	3.5.2	4
OpenSSL	3.0.13	3
wolfSSL	5.7.0	264
Go stdlib	1.22.1	264
cryptography	42.0.5	3

(c) Number of unique certificates that were classified with “x509: invalid ECDSA parameter”.

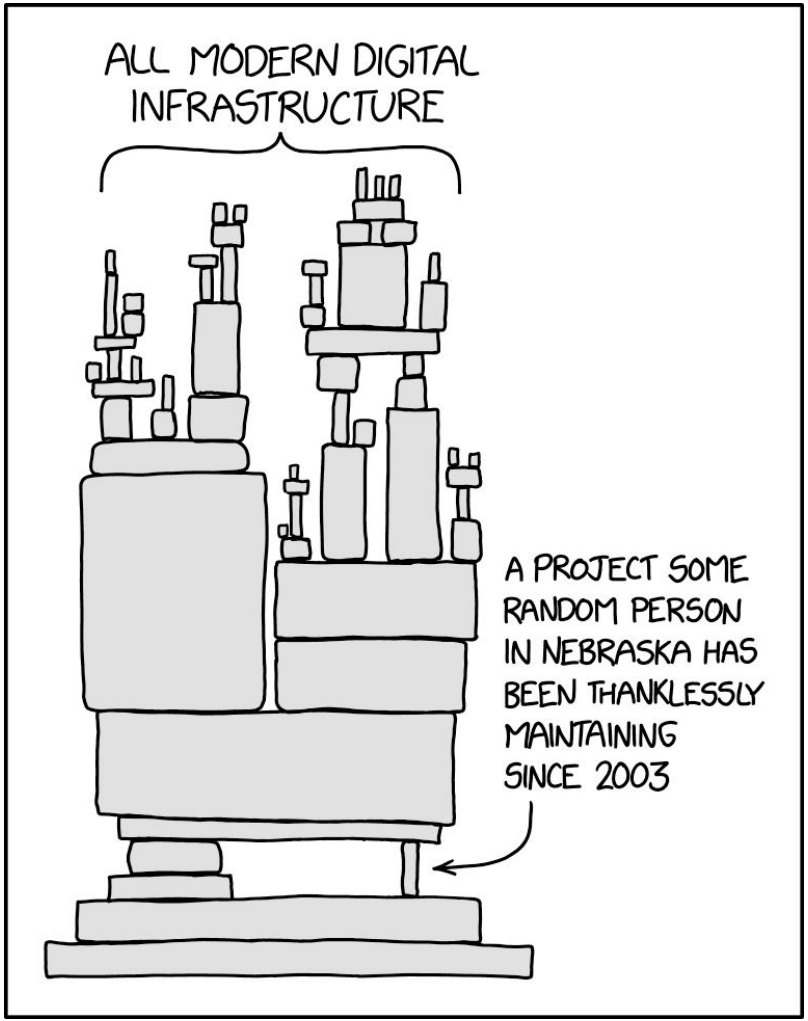
Library	Version	Count
GnuTLS	3.8.1	0
Mbed TLS	3.5.2	17
OpenSSL	3.0.13	0
wolfSSL	**5.5.4**	17
wolfSSL	**5.7.0**	3
Go stdlib	1.22.1	17
cryptography	42.0.5	0

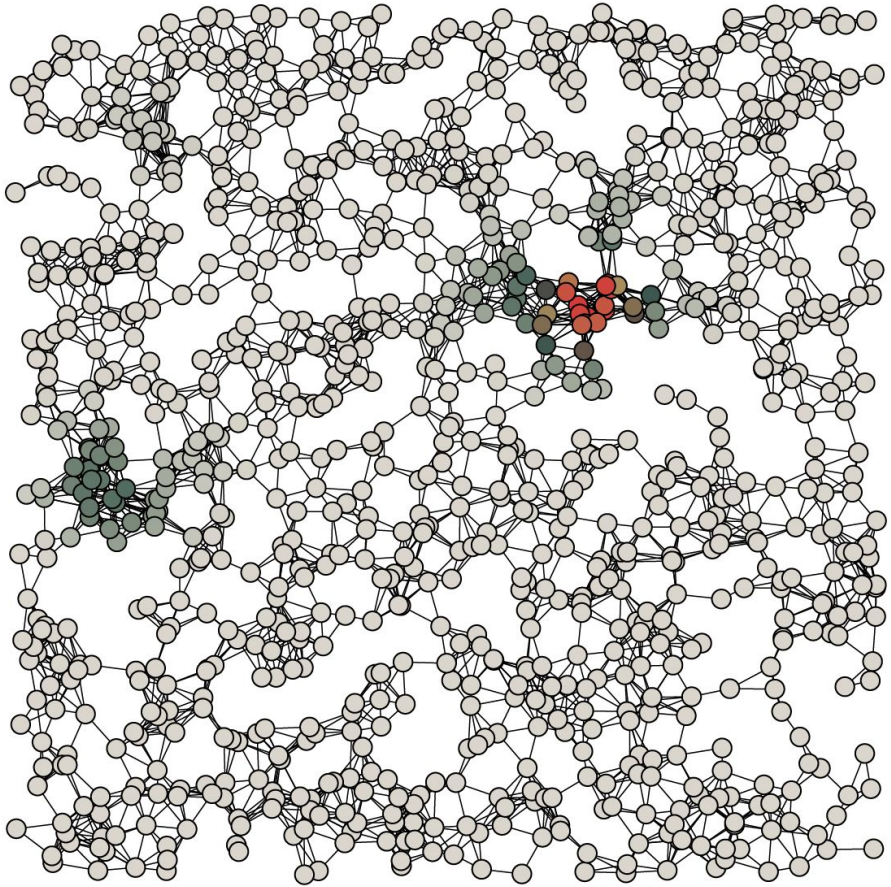
Impact of Vaguely Written Specifications

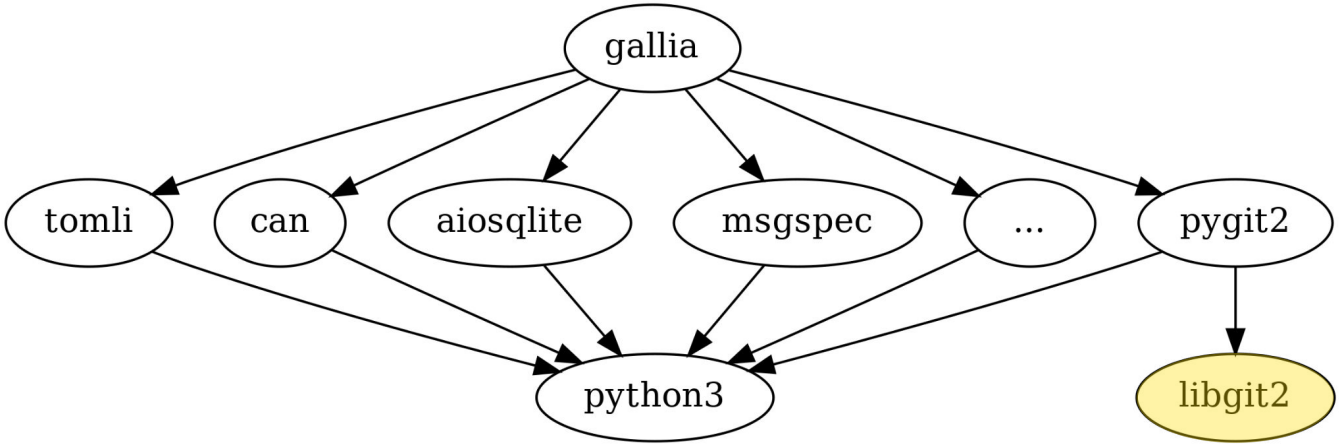
Complex and vaguely written specifications lead to implementations with different design and runtime behaviour. This might be a risk for the overall ecosystem (cf. DOI: 10.14722/ndss.2023.23072).

$$\text{expected cost} = \sum_{b \in \text{bad outcomes}} \text{cost}(b) \cdot \text{probability}(b)$$

https://doi.org/10.1145/3347446






gallia
tomli
can
aiosqlite
msgspec
...
pygit2
python3
libgit2

Excerpt of Important Software Modules (in C)

1. **python3** (Programming Language)
2. **zlib** (Compression)
3. **texinfo** (Text Processing)
4. **emacs** (Text Editor)
5. **openssl** (Cryptography) 
6. **ncurses** (Terminal)
7. **bash** (Shell) 
8. **libxcrypt** (Cryptography)
9. **expat** (XML)
10. **bzip2** (Compression)
11. **libffi** (Foreign Function Interface)
12. **perl** (Programming Language)
13. **xz** (Compression) 
14. …

Mentionable Effort

Finding Anomalies in the Debian Packaging System to Detect Supply Chain Attacks
Tobias Specht `@FOSDEM2025` (https://fosdem.org/2025/)

Detecting Supply Chain Attacks from the Filesystem Level with eBPF, fanotify and others
Tobias Specht `@ALPSS2025` (https://www.alpss.at/2025)

`#source-build-tracing:matrix.org`

Conclusions: What are current challenges?

- Impact of Vague Specifications
  - Complex and vague specifications lead to implementations with different design and runtime behaviour (cf. DOI: 10.14722/ndss.2023.23072).
  - FLOSS ecosystem as a whole seems to be fragile in some aspects.
- Pinpoint Issues in Software Ecosystems
  - Available algorithms can be used to pinpoint problematic parts.
  - No standardized maintenance and vulnerability tracking available, especially for non Github repositories.
    - SBOMs help, but we need them as a first class citizen in package tooling!
    - Marius Biebel, *Quality Assessment of SBOM Generation Tools and Standards on Open Source Projects*, Ma-Thesis https://mariuxdeangelo.de/assets/docs/Masterthesis.pdf
  - Sustainability can be estimated on metrics such as: **bus factor, license, number of (reverse) dependencies, number of CVEs**, or especially **funding**.

Research Conclusions

In order to improve our software architecture sustainability, we need to…

- …improve standardization processes.
- …find and address insufficiently maintained projects.
- …better pool resources.
- …teach industry partners use the available tooling.

Credits

The Current Role of Memory-safe Programming Languages in the Real World (2025, Ba-Thesis)
Florian Steinbauer: https://www.linkedin.com/in/florian-steinbauer-0a649318b/

Dynamic and Automatic Validation of Security Mechanisms for QUIC (2024, Ma-Thesis)
Nguyen Truong An To: https://www.linkedin.com/in/nguyen-truong-an-to-40a3351b8/

Not Quite As QUICk: Automatic Vulnerability and RFC Security Considerations Compliance Testing of QUIC Protocol Server Implementations (2025, Ba-Thesis)
Kacper Darowski: https://www.linkedin.com/in/kacper-darowski/

ELISA
Enabling Linux in Safety Applications

WORKSHOP

Dr. Stefan Tatschner

@rumpelsepp
in/stefan-tatschner
0000-0002-2288-9010

ALPAKA

ALPAKA — Vernetzung und Sicherheit digitaler Systeme