Empirical Study of Memory Bugs in Database Systems

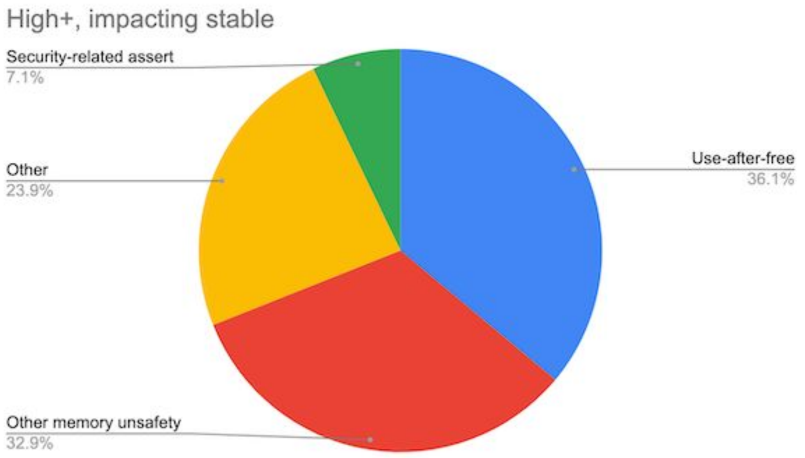
Lan Ouyang
Advisor: Martin Fink,
Ilya Meignan--Masson
Chair of Computer Systems
https://dse.in.tum.de/

07.08.2024 – 07.02.2025

Motivation

- Around 70% of serious security bugs in Google Chromium project[1] and Microsoft projects[2] are memory safety problems.

- Database systems are particularly vulnerable due to the widespread use of memory-unsafe languages like C and C++.

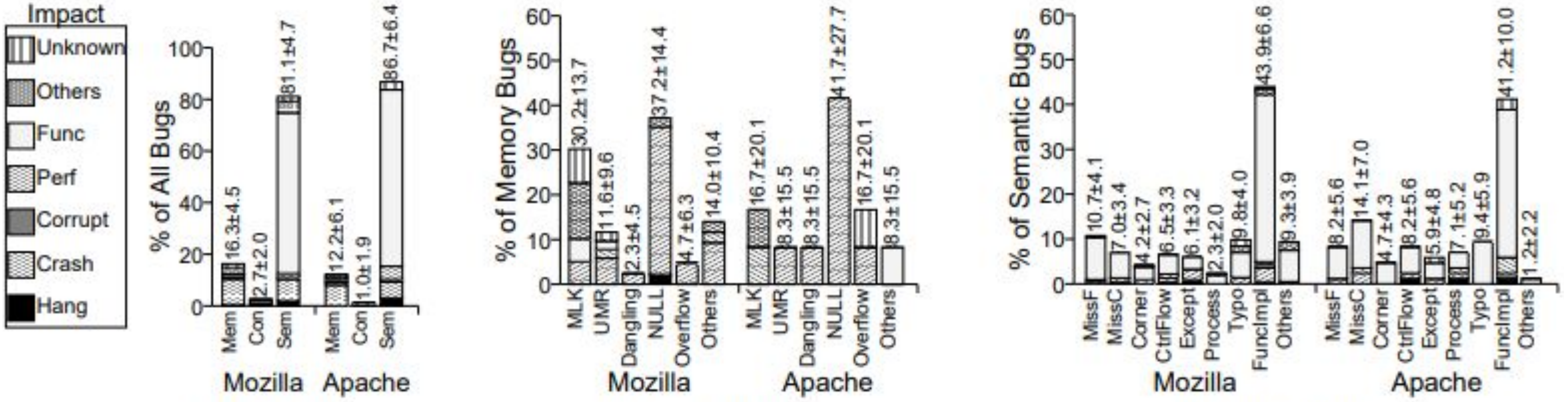


Chromium: Analysis based on 912 high or critical severity security bugs since 2015[1]

[1] Memory safety
[2] A proactive approach to more secure code | MSRC Blog | Microsoft Security Response Center

State-of-the-art

Li et al. [3] investigate the evolving nature of software bugs in modern open-source software by analyzing fixed runtime bugs from Mozilla and Apache HTTP Server.



(a) All bugs (b) Memory bugs (c) Semantic bugs

Figure 1: Distribution of root causes with impacts. The numbers show 95% confidence level.

[3] Z. Li, L. Tan, X. Wang, S. Lu, Y. Zhou, and C. Zhai. “Have things changed now? An empirical study of bug characteristics in modern open source software.” In Proceedings of the 1st workshop on Architectural and system support for improving software dependability. 2006, pp. 25–33.

State-of-the-art

Zhang et al.[4] study bugs and fixes by analyzing 1100 bug-fix commits across six major Rust projects and the Rust standard library.

TABLE II: Root causes of bugs in our dataset.

Project \ Root Cause	Alg	Cod	Mem	Type	Conc	Bound	Doc	Num	Unw	Owner	Attr	Mac	Unsound	Ver	Others	**Total**
Rust-analyzer	160	2	0	1	0	7	45	0	9	1	0	1	0	3	1	**230**
Egui	38	0	0	2	3	0	1	2	4	1	4	3	0	8	0	**66**
Tikv	143	14	24	6	36	3	4	1	26	6	6	14	1	10	0	**294**
Tokio	25	0	3	6	6	1	21	1	3	1	4	5	2	4	0	**82**
Meilisearch	30	7	0	0	2	1	14	0	3	0	4	0	0	5	2	**68**
SnarkOS	17	0	1	0	0	0	2	0	3	0	1	2	4	0	0	**30**
std-lib	4	0	2	0	1	1	4	1	1	1	1	1	2	1	0	**20**
Total	**417**	**23**	**30**	**15**	**48**	**13**	**91**	**6**	**49**	**10**	**20**	**26**	**5**	**35**	**3**	**790**

[4] C. Zhang, Y. Feng, Y. Zhang, Y. Dai, and B. Xu. "Beyond Memory Safety: an Empirical Study on Bugs and Fixes of Rust Programs." In: 2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS). IEEE. 2024, pp. 272–283.

Research gap

Prior empirical studies of bugs

- No database-specific analysis

- Limited dataset scope

- Focus on generic bug types

Problem statement

To understand the impact of memory safety bugs on database systems, we investigate all of the reported bugs of 7 databases.

Problem statement

- RQ1: How often do memory bugs appear in database systems?
- RQ2: What kind of memory bugs appear in database systems?
- RQ3: Are memory bugs diminishing in database systems in the past years?
- RQ4: How severe are various kinds of memory bugs?
- RQ5: How urgent are memory bugs in database systems?
- RQ6: How are memory bugs distributed in different components in the database system?
- RQ7: How many memory bugs in database systems can be exploited to compromise security?

Methodology

- Bug collection
  - API
  - Web scraping
  - CVEs downloaded from CVE: Common Vulnerabilities and Exposures
- Keyword-based memory bug identification
- Irrelevant bug report removal
- Analysis dimensions
  - bug type
  - date of creation
  - impact
  - software component

Methodology

Table 3.1.: Memory bug types and keywords

Type	Keyword
Memory Leak	memory leak
Over-/underflow	overflow buffer overflow buffer overrun buffer underflow buffer underrun out of bounds array bounds stack overflow stack smashing integer overflow
Null Pointer Dereference	null pointer dereference null pointer exception
Double Free	double free multiple free
Uninitialized Memory	uninitialized memory use of uninitialized wild pointer
Dangling Pointer	dangling pointer dangling reference use after free UAF stack use after scope stack use after return
Others	memory safety memory corruption heap corruption stack corruption memory error invalid memory access illegal memory access segmentation fault SIGSEGV address sanitizer ASLR

Implementation

- Data scraping
  - Scrapy
  - Jira & Github API
  - rate limiting
- Elasticsearch storage
  - field data type mapping
- Elasticsearch query
  - Full text search
  - aggregation

Evaluation

RQ1&2: Memory bug frequency and types

MariaDB, MySQL, and Redis have the most memory bugs reported.

Over-/underflow, memory leak, and dangling pointer are the most frequent.

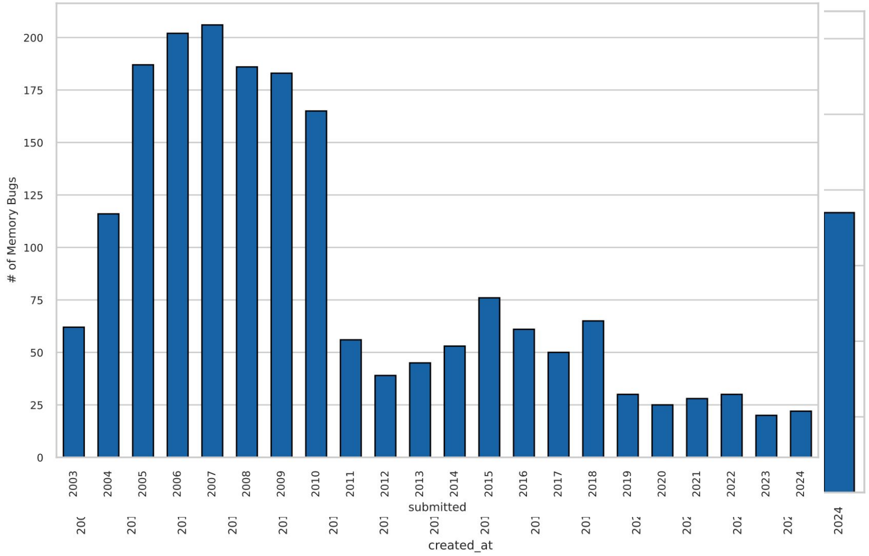
Table 5.2.: Number of bugs per type in 7 databases

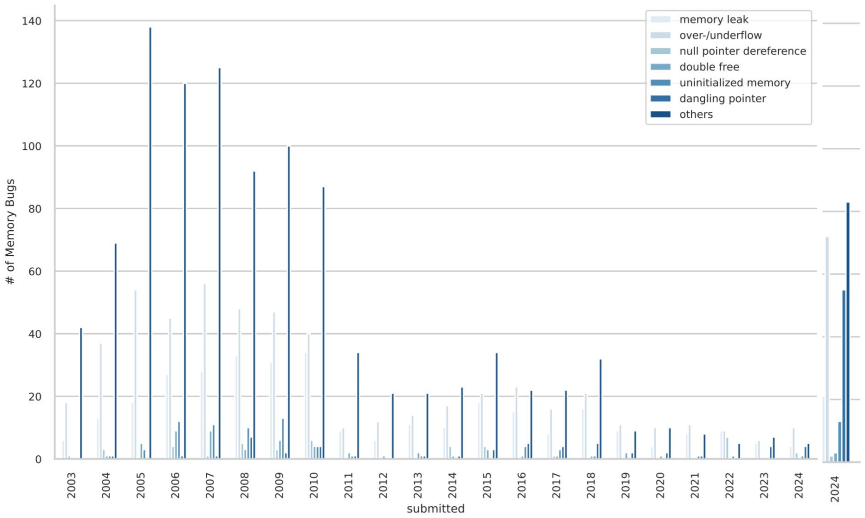
Database	ML	ODF	NPD	DF	UM	DP	Others	All memory bugs	Total
MySQL	322	536	41	52	67	49	1026	1907	43706
SQLite	2	11	1	0	0	4	5	23	624
MariaDB	228	810	18	62	113	596	872	2076	24527
Redis	143	235	25	22	3	28	480	827	11838
LevelDB	11	11	1	0	0	2	18	42	1056
DuckDB	1	3	0	1	0	1	10	16	488
RocksDB	81	85	1	12	3	57	209	413	12266

Evaluation

RQ3: Memory bug trends

Memory bugs are not diminishing in most databases, and even rising in MariaDB.





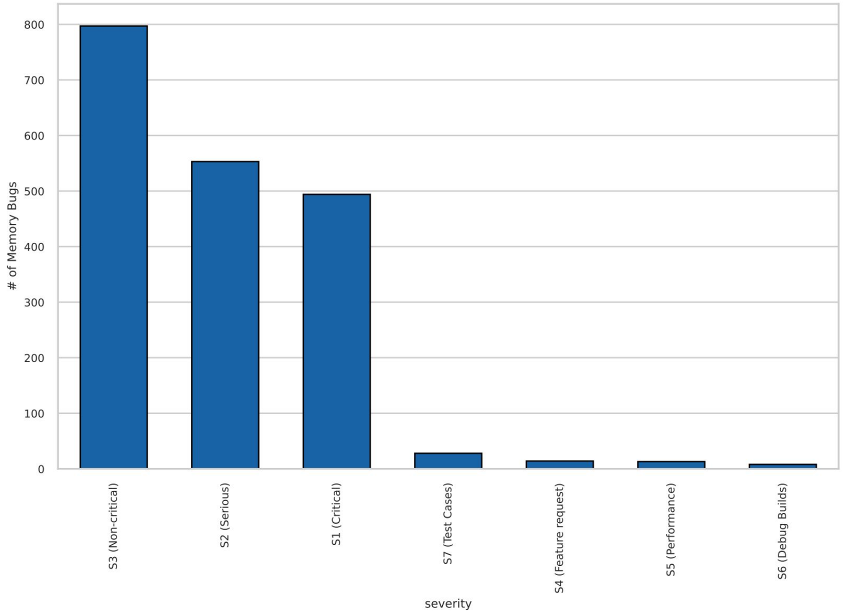
Number of memory bugs from MySQL per year

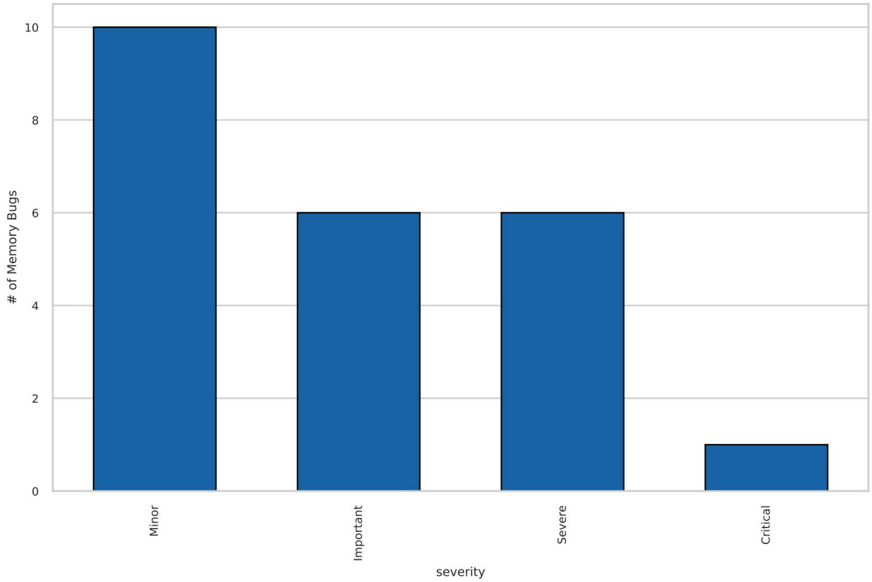
Number of memory bugs from MariaDB per year

Evaluation

RQ4: Memory bug severity

Over half of the memory bugs are labeled as at least serious or important.



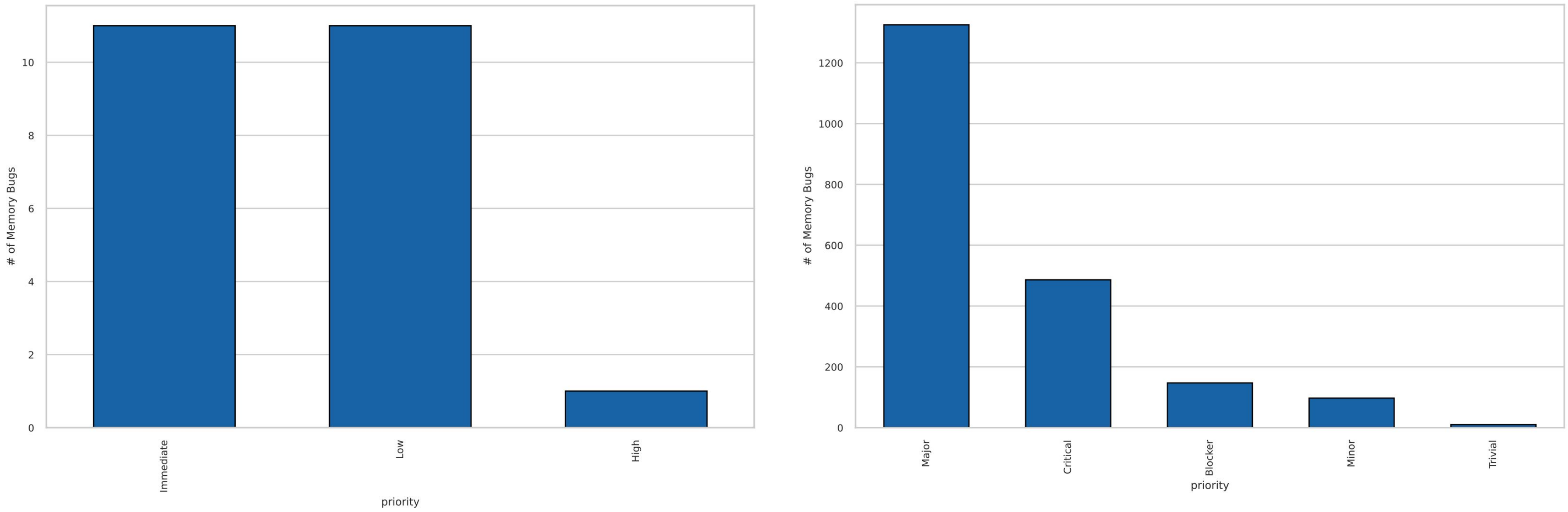


Number of memory bugs from MySQL (left) and SQLite (right) per severity

Evaluation

RQ5: Memory bug priority

Most of the memory bugs in SQLite and MariaDB are of high priority

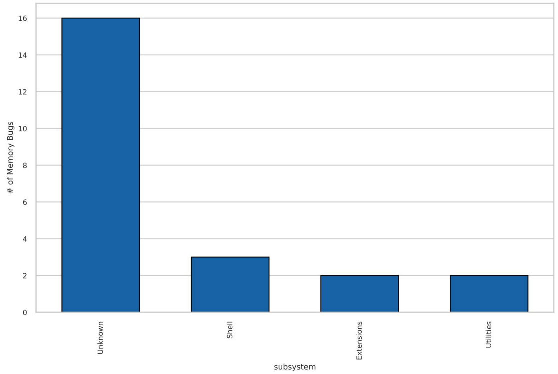


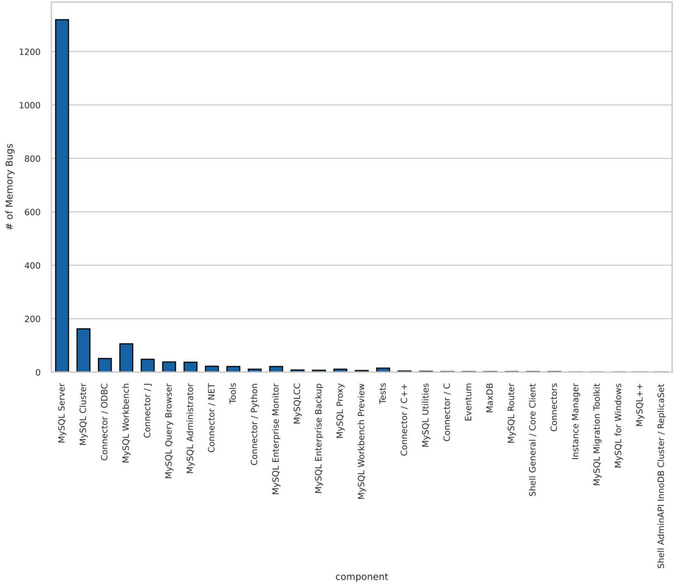
Number of memory bugs from SQLite (left) and MariaDB (right) per priority

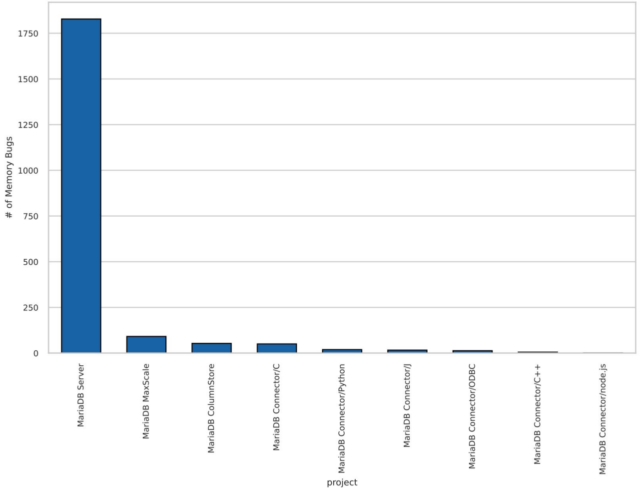
Evaluation

RQ6: Memory bug locations

Most of the memory bugs in MySQL and MariaDB are in database server







Number of memory bugs from SQLite (left) , MySQL (middle), and MariaDB (right) per location

Evaluation

RQ7: Memory bug caused security exploitations

There are higher proportion of memory bugs found in CVEs than in bug tracking systems

Table 5.3.: Number of CVE records per type per database

Database	ML	ODF	NPD	DF	UM	DP	Others	All Memory Bugs	Total
MySQL	0	0	0	0	0	1	0	1	1022
SQLite	0	24	1	0	2	1	16	44	241
MariaDB	0	9	0	0	0	0	16	25	203
Redis	1	38	0	0	0	1	1	41	201
LevelDB	0	0	0	0	0	0	0	0	1
DuckDB	0	0	0	0	0	0	0	0	3
RocksDB	0	0	0	0	0	1	0	1	4

Table 5.5.: Number of security vulnerabilities per type from Redis

Bug type	ML	ODF	NPD	DF	UM	DP	Others	All Memory Bugs	Total
Count	0	22	1	0	0	0	0	23	32

Table 5.6.: Number of security vulnerabilities per type from SQLite

Bug type	ML	ODF	NPD	DF	UM	DP	Others	All Memory Bugs	Total
Count	0	3	6	0	5	5	0	19	24

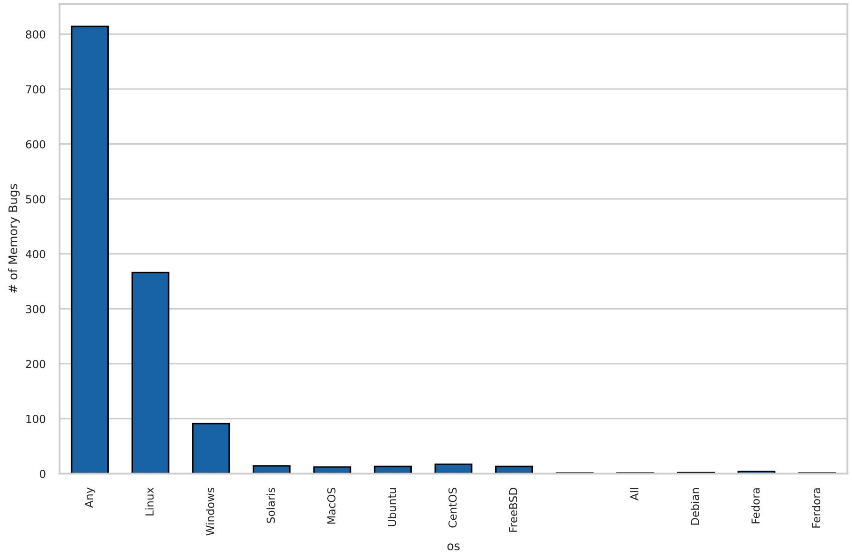
Table 5.4.: Number of CVE records reported as a bug per type per database

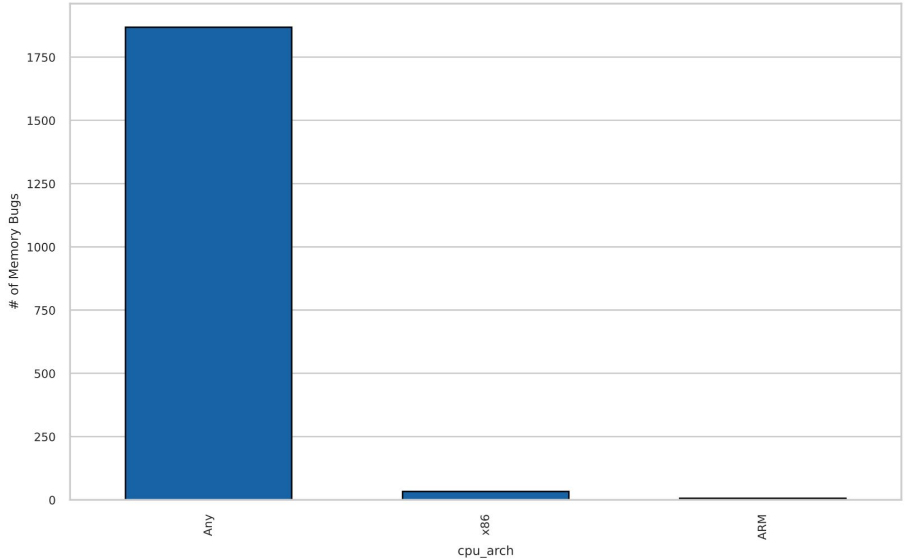
Database	ML	ODF	NPD	DF	UM	DP	Others	All
MariaDB	0	2	0	0	0	0	15	17

Evaluation

Additional analysis: operating system and CPU arch

Most memory bugs are reported on Unix/Linux systems, excluding unspecified





Number of memory bugs from MySQL per os (left) and per cpu arch (right)

Summary

- Existing empirical bug studies are not focused on databases or memory safety

- Large-scale empirical study of memory safety bugs in databases
  - Critical insights into the nature, trends, and impacts of memory safety bugs
  - A reproducible framework for collecting and analyzing memory bugs across diverse systems

- Memory safety is a persistent threat in databases