Static Analysis of Redis using Cppcheck

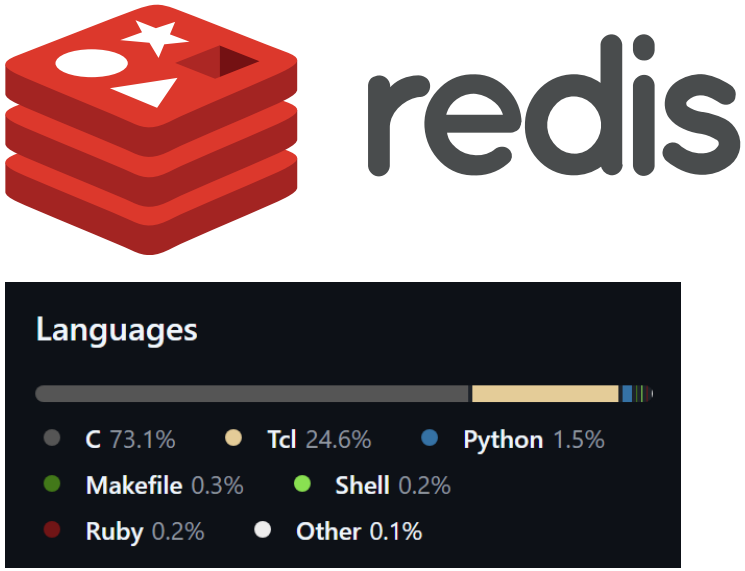
Radu-Constantin Onuțu

University of Bucharest

Table of Contents

Repository Overview: **Redis**

- **Redis** = REmote DIctionary Server
- In-memory, key–value NoSQL DB with sub-millisecond latency
- Core written in `C`
- 69.3k stars on GitHub, 753 contributors
- First released 2009

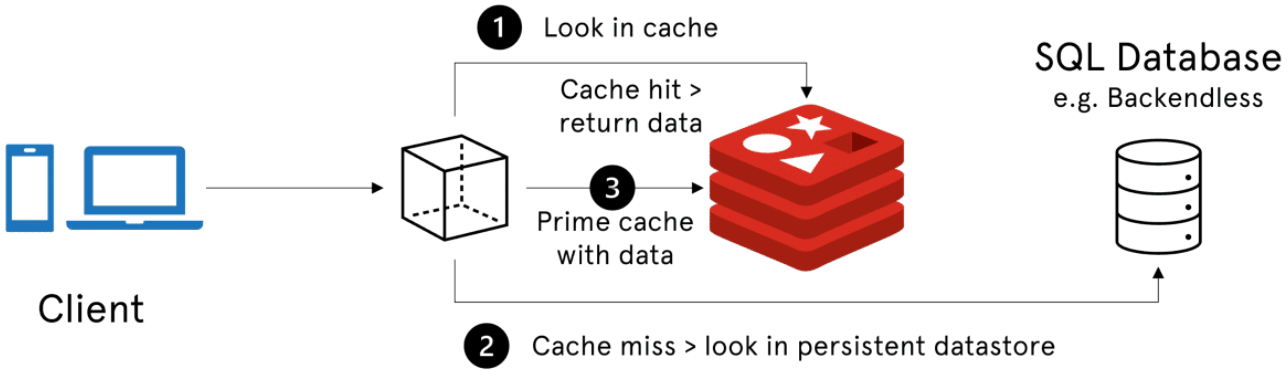


Repository overview: Redis

Rich data structures:

- Strings
- Hashes
- Lists
- Sets
- Sorted sets
- Vector sets
- Streams
- Bitmaps
- Bitfields
- Geospatials
- JSONs
- Probabilistic data types
- Time series

How Redis is typically used



Repository Overview: **Redis**

Rank May 2025	Rank Apr 2025	Rank May 2024	DBMS	Database Model	Score May 2025	Score Apr 2025	Score May 2024
1.	1.	1.	Redis	Key-value, Multi-model	152.19	-1.92	-5.61
2.	2.	2.	Amazon DynamoDB	Multi-model	79.30	+2.81	+5.23
3.	3.	3.	Microsoft Azure Cosmos DB	Multi-model	22.94	+0.39	-6.10
4.	4.	4.	Memcached	Key-value	15.47	+0.49	-3.95
5.	5.	5.	etcd	Key-value	7.14	+0.09	-0.11
6.	6.	6.	Hazelcast	Key-value, Multi-model	5.58	+0.03	-0.39
7.	7.	7.	Aerospike	Multi-model	5.45	+0.22	-0.32
8.	8.	8.	Ehcache	Key-value	3.98	+0.04	-0.91
9.	9.	↑ 17.	Oracle NoSQL	Multi-model	3.39	-0.04	+0.44
10.	↑ 12.	↑ 15.	Apache Ignite	Multi-model	3.23	+0.13	+0.07

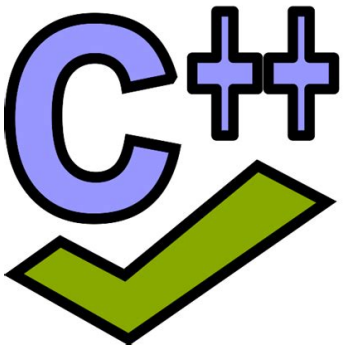
The world's most popular **NoSQL** database.

Rank May 2025	Rank Apr 2025	Rank May 2024	DBMS	Database Model	Score May 2025	Score Apr 2025	Score May 2024
1.	1.	1.	Oracle	Relational, Multi-model	1226.57	-4.49	-9.72
2.	2.	2.	MySQL	Relational, Multi-model	964.98	-22.12	-118.76
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	774.89	-10.12	-49.41
4.	4.	4.	PostgreSQL	Relational, Multi-model	674.32	+7.07	+28.77
5.	5.	5.	MongoDB	Document, Multi-model	402.51	+2.46	-19.14
6.	6.	↑ 9.	Snowflake	Relational	172.01	+3.92	+50.67
7.	7.	↓ 6.	Redis	Key-value, Multi-model	152.19	-1.92	-5.61
8.	↑ 9.	8.	IBM Db2	Relational, Multi-model	126.40	+0.36	-2.06
9.	↓ 8.	↓ 7.	Elasticsearch	Multi-model	123.81	-4.27	-11.54
10.	10.	10.	SQLite	Relational	117.77	+3.68	+3.45

Ranked **7-th** among most popular databases.

Linter overview: Cppcheck

- **Purpose**: find real defects in C/C++, minimise false positives
- **Release**: 2007 by Daniel Marjamäki
- **Cross-platform**: runs on Windows, Linux, macOS
- **Outputs**: CLI, XML/JSON, GUI
- Offers a GUI interface, useful for less-experienced users



Linter overview: Cppcheck

Cppcheck 2.17.1 - C:\Users\Radu\Downloads\redis-unstable\redis-unstable

File Edit View Analyze Help

Quick Filter:

69% (352 of 442 files checked)

There was a critical error with id 'preprocessorErrorDirective', when checking C:/Users/Radu/Downloads/redis-unstable/redis-unstable/src/redis-check-rdb.c. Analysis was aborted.

File	Line	Severity	Summary	Id	CWE
deps\fast_float\fast_float_strtod.cpp					
deps\fast_float\fast_float_strtod.cpp	0	inform...	Too many #ifdef configurations - cppcheck only checks 12 of 36 configurations. Use --for...	toomanyconfigs	398
deps\fast_float\fast_float_strtod.cpp	2	inform...	Include file: <iostream> not found. Please note: Cppcheck does not need standard library...	missingIncludeSystem	0
deps\fast_float\fast_float_strtod.cpp	3	inform...	Include file: <string> not found. Please note: Cppcheck does not need standard library he...	missingIncludeSystem	0
deps\fast_float\fast_float_strtod.cpp	4	inform...	Include file: <system_error> not found. Please note: Cppcheck does not need standard lib...	missingIncludeSystem	0
deps\fast_float\fast_float_strtod.cpp	5	inform...	Include file: <cerrno> not found. Please note: Cppcheck does not need standard library h...	missingIncludeSystem	0
deps\fast_float\fast_float_strtod.cpp	0	inform...	Limiting analysis of branches. Use --check-level=exhaustive to analyze all branches.	normalCheckLevelMaxBranches	0
deps\fast_float\fast_float_strtod.cpp	29	style	C-style pointer casting	cstyleCast	398
deps\fast_float\fast_float.h					
deps\fast_float\fast_float.h	92	inform...	Include file: <cfloat> not found. Please note: Cppcheck does not need standard library he...	missingIncludeSystem	0
deps\fast_float\fast_float.h	93	inform...	Include file: <cstdint> not found. Please note: Cppcheck does not need standard library h...	missingIncludeSystem	0

Id: va_list_usedBeforeStarted
CWE: 664
va_list '_cpy' used before va_start() was called.

```
389                 va_list _cpy;
390
391                 /* Flags */
392                 while (*_p != '\0' && strchr(flags,*_p) != NULL) _p++;
393
394                 /* Field width */
395                 while (*_p != '\0' && isdigit((int) *_p)) _p++;
396
397                 /* Precision */
398                 if (*_p == '.') {
399                     _p++;
400                     while (*_p != '\0' && isdigit((int) *_p)) _p++;
401                 }
402
403                 /* Copy va_list before consuming with va_arg */
404                 va_copy(_cpy,ap);
405
406                 /* Make sure we have more characters otherwise strchr() accepts
407                  * '\0' as an integer specifier. This is checked after above
408                  * va_copy() to avoid UB in fmt_invalid's call to va_end(). */
409                 if (*_p == '\0') goto fmt_invalid;
410
411                 /* Integer conversion (without modifiers) */
412                 if (strchr(intfmts,*_p) != NULL) {
413                     va_arg(ap,int);
414                     goto fmt_valid;
415                 }
416
417                 /* Double conversion (without modifiers) */
418                 if (strchr("eEfFgGaA",*_p) != NULL) {
419                     va_arg(ap,double);
420                     goto fmt_valid;
```

Analysis Log | Warning Details

Linter overview: Cppcheck

Running Cppcheck over Redis:

Errors : 81

Warnings : 135

Style warnings : 1505

Portability warnings : 4

Performance warnings : 0

Information messages : 1732

Out of 81 errors, 33 (40.74%) were false positives (many of them due to unknown macros that were, in fact, correctly defined).

Bug #1 – CWE-401: uninitialised va_list

Bug summary

Error: CWE 401 ("va_list _cpy used before va_start()"). A va_list becomes valid only after initialisation with va_start or va_copy. In the original code the copy is made unconditionally; an early return may exit the function before _cpy receives a matching va_end(), yielding undefined behaviour.

Before

```
1 va_list _cpy;
```

After

```
1 va_list _cpy;
2 va_copy(_cpy, ap);
```

Bug #2 – CWE-457: uninitialised struct member `c.err`

Bug summary

Error: CWE 457 ("Uninitialised struct member: `c.err`"). Inside `test_invalid_timeout_errors()` the pointer `redisContext *c` is left *unset* when the connection type falls through the `else` branch. Execution then reaches code that reads `c->err`, producing undefined behaviour and a spurious "Invalid timeout specified" message.

Before

```
if (...) {
    c = redisConnectWithTimeout
        (...);
} else if (...) {
    c =
        redisConnectUnixWithTimeout
        (...);
} else {
    assert(NULL); /* c remains
        NULL */
}
```

After

```
if (...) {...} else if (...)
    {...}
else {
    fprintf(stderr, "
        Unsupported_connection␣
        type_%d\n", config.type);
    abort(); /* never reach
        invalid read */
}
```

Bug #3 – CWE-457: uninitialised struct member node

Bug summary

Error: CWE 457 ("Uninitialised struct member: `node`"). The macro `RTREE_GET_CHILD(0)` assumes that the local pointer `node` already references the radix-tree root. In the original code `node` is declared but never initialised before the macro is expanded, so the first statement inside the macro dereferences an indeterminate struct, triggering undefined behaviour.

Before

```
1 if (RTREE_HEIGHT > 1) {
2     RTREE_GET_CHILD(0)
3 }
```

After

```
1 if (RTREE_HEIGHT > 1) {
2     rtree_node_elm_t *node =
3         rtree_root_node_read(tsdn,
              rtree, dependent);
4     if (node == NULL) {
5         return NULL; /* fail fast
              */
6     }
7     ctx->child[0] = node; /*
          initialise */
8     RTREE_GET_CHILD(0)
9 }
```

Bug #4 – CWE-401: memory leak (p not freed)

Bug summary

Error: CWE 401 ("Memory leak: p"). The test allocates a buffer with `aligned_alloc()`, performs a few alignment/contents checks, and then returns without releasing the memory. Because the pointer `p` is never passed to `free()`, the allocation is lost.

Before

```
p = aligned_alloc(alignment, size
    );
expect_false(p != NULL ||
    get_errno() != ENOMEM,
   "Expected error for
       aligned_alloc(%zu, %zu)",
   alignment, size);
```

After

```
p = aligned_alloc(alignment, size
    );
expect_false(p != NULL ||
    get_errno() != ENOMEM,
   "Expected error for
       aligned_alloc(%zu, %zu)",
   alignment, size);
free(p); /* release allocation */
```

Bug #5 – CWE-401: lost pointer after `realloc()` failure

Bug summary

Error: CWE 401 ("Common `realloc` mistake: `p` nulled but not freed upon failure"). When `realloc(p, large0)` is assigned straight back to `p`, a failure (returning `NULL`) destroys the only reference to the original block, causing an unavoidable leak. The fix copies the result to a temporary, checks it, then updates `p` and frees the allocation on exit.

Before

```
p = realloc(p, large0);
```

After

```
void *newp = realloc(p, large0);
assert_ptr_not_null(newp, "
    realloc()␣failed");
p = newp; /* update only if
    realloc succeeded */
```

Bug #6 – CWE-682: pointer arithmetic on NULL

Bug summary

Error: CWE 682 ("If memory allocation fails: pointer addition with NULL pointer"). The variable `elem_iter` is assigned the result of `calloc()`. If the allocation were to fail and return NULL, the subsequent pointer arithmetic (`elem_iter += ...`) performs undefined behaviour, potentially crashing the test. The fix verifies the pointer before any use.

Before

```
elem_t *elems = calloc(...);
elem_t *elem_iter = elems;
for (...) {
    ...
    elem_iter += NELEMS_PER_PUSHER
        ; /* UB if elem_iter is
        NULL */
}
```

After

```
elem_t *elems = calloc(...);
elem_t *elem_iter = elems;
assert_ptr_not_null(elem_iter, "
    calloc() failed in mpsc_queue
    test");
for (int i = 0; i < NPUSHERS; i
    ++) {
    ...
    elem_iter += NELEMS_PER_PUSHER
        ; /* safe: non-NULL */
}
```

Thank you!

References

- github.com/redis/redis – Redis repository
- redis.io/docs – Redis documentation
- github.com/danmar/cppcheck – Cppcheck repository
- db-engines.com/en/ranking/key-value+store - DB-Engines ranking: key–value stores
- db-engines.com/en/ranking - DB-Engines overall database ranking