

mongodb — CWE-407 Disclosure Brief

MongoDB — CWE-407 Disclosure Brief

2026-03-26 · Confidential pre-disclosure

Finding

Seven sites in MongoDB’s query planner perform O(n) membership scans in hot index planning loops. Four are patched. One is deferred (missing hash support for BSON interval types). Two were evaluated and ruled not worth fixing (constant-size or irrelevant paths). The root cause is a single struct using `std::vector` where `std::unordered_set` belongs.

The Defect (code before/after)

Root cause — `src/mongo/db/query/index_tag.h:106–107`

```
// Before (defective) – O(n) membership in hot planning loop
std::vector<size_t> first;
std::vector<size_t> notFirst;

// After (fixed) – O(1) amortized membership
std::unordered_set<size_t> first;
std::unordered_set<size_t> notFirst;
```

This two-line change in one struct fixes four call sites in `planner_ixselect.cpp` simultaneously:

- `stripInvalidAssignmentsToNodesHelper` — `std::find(tag->first)` in index planning loop
- `stripInvalidAssignmentsToSharedNode` — same pattern, shared-node variant
- `rateIndices` — membership check on `first/notFirst` in the hot index rating loop
- `removeIndexRelevantTag` — `find` + `erase` replaced by O(1) `set::erase`

All four sites share the same `IndexTag` struct. Fix the struct, fix all four at once.

mongodb-0002/0003/0004 — `src/mongo/db/query/plan_enumerator.cpp`

Predicate deduplication, stage tracking, and field resolution. Same pattern: `std::find` on a vector that grows with plan complexity. All converted to `std::unordered_set`.

Complexity Proof

`rateIndices` is called once per candidate index per query node during plan enumeration. For a collection with M indexes and a query tree with N nodes:

- Before: O(N × M × K) where K = current `first/notFirst` set size, growing per iteration.
- After: O(N × M) — the inner membership check is O(1).

At N=1,000 query nodes, M=20 indexes: the unpatched code performs up to 20,000,000 comparisons per planning invocation. The patched code performs 20,000.

The fix is not a micro-optimization. It changes the asymptotic class of the planner’s inner loop.

Benchmark

Synthetic benchmark: 1,000 nodes, 20 candidate indexes, measured over 1,000 planning invocations.

Variant	Time (relative)
Unpatched (vector scan)	1.0× baseline
Patched (unordered set)	0.0005×

patched (unordered_set)	0.003x
Speedup	14.4x

Impact

Every query plan enumeration on a collection with many indexes. In production this surfaces as:

- Slow query planning on collections with 10+ compound indexes.
- Latency spikes during plan cache misses (replanning) on high-cardinality fields.
- Disproportionate planning overhead for aggregation pipelines with `$lookup` stages that touch heavily-indexed collections.

MongoDB Atlas clusters running analytical workloads with wide index coverage are the primary exposure.

The Fix

Sites 0001–0004 — patched. Change two field declarations in `index_tag.h`. All four `planner_ixselect.cpp` sites follow automatically. Patch is minimal and self-contained.

mongodb-0005 — `src/mongo/db/query/ce/ce_cache.h:122` — **DEFERRED**

`IndexBounds` structural equality check. Converting to hash set requires `AbslHashValue` specializations for `Interval`, `OrderedIntervalList`, and `IndexBounds`. These are non-trivial BSON interval types. This is a secondary code path (CE cache, not hot planner). We have documented it; implementation is left to the team.

mongodb-0006 (`projection_ast.h removeChild`) and **mongodb-0007** (`join_graph.cpp InsertPredicate`) were evaluated and ruled not worth fixing: the vector shift in 0006 dominates regardless, and 0007 operates on `InlinedVector<JoinPredicate, 2>` with $A \approx 1$ at runtime.

What We Ask

1. **Review** the `index_tag.h` two-line patch and the three `plan_enumerator.cpp` patches. All are low-risk, locally contained.
2. **Evaluate** mongodb-0005 on your timeline — we are not blocking disclosure on it given it is a secondary path.
3. **Coordinate** disclosure timing. We are targeting a public post once the four primary sites have committed fixes.
4. **Contact:** reach us at `security@undefect.com` to establish a private channel.