

presto — CWE-407 Disclosure Brief

Presto — CWE-407 Disclosure Brief

2026-03-26 · Patch available — awaiting upstream merge

Finding

Four $O(n^2)$ defects in Presto's SQL query optimizer. All patched. Patches ready for upstream review. Three defects are in `PushDownDereferences.java` — the dereference pushdown optimizer rules applied during query optimization. One is in `PayloadJoinOptimizer.java`.

The Defects

presto-0001 (PATCHED — MEDIUM): `presto-main-`

`base/src/main/java/com/facebook/presto/sql/planner/iterative/rule/PushDownDereferences.java:206`

```
// Inside JoinNodeRewriteRule.rewrite():
for (Map.Entry<...> entry : expressions.inverse().entrySet()) {
    VariableReferenceExpression baseVariable = getBase(entry.getValue());
    if (joinNode.getLeft().getOutputVariables().contains(baseVariable)) { // O(V)
        leftSideDereferences.put(...);
    }
}
```

`getOutputVariables()` returns an `ImmutableList`. `ImmutableList.contains()` is a linear scan. Called D times (once per dereference expression): **$O(D \times V)$ per rule invocation.**

presto-0002 (PATCHED — MEDIUM): `PushDownDereferences.java:369` — identical pattern in a second `JoinNode` pushdown rule.

presto-0003 (PATCHED — MEDIUM): `PushDownDereferences.java:414` — identical pattern in the `SemiJoinNode` pushdown rule.

presto-0004 (PATCHED — MEDIUM): `presto-main-`

`base/src/main/java/com/facebook/presto/sql/planner/optimizations/PayloadJoinOptimizer.java:208`

```
// Inside rewrite():
ImmutableSet<VariableReferenceExpression> rightJoinKeys = inputJoinKeys.stream()
    .filter(key -> rightNode.getOutputVariables().contains(key)) // O(V) per key
    .collect(toImmutableSet());
```

`getOutputVariables()` called K times (once per join key): **$O(K \times V)$.**

Complexity Proof

For D dereference expressions and V output variables (worst case: no match, full scan): - Each

`ImmutableList.contains()` scans V entries - Called D times per rule invocation - Total: **$O(D \times V)$**

At D=V=100 (worst case, all misses): defective=10,000 comparisons, fixed=100. **Measured ratio: 100×**.

The Presto optimizer runs these rules on every query containing dereference expressions (field access on row types, struct projections, nested column access). Complex analytical queries with many output columns from wide row types maximize $D \times V$.

Impact

Presto is the distributed SQL engine used at Meta, Uber, Airbnb, Netflix, Twitter, and LinkedIn for interactive analytics. Queries over wide row types (e.g., nested JSON columns, Hive struct fields, Iceberg nested schemas) are common in data lake workloads. These pushdown rules run during query optimization for every such query. The

overhead scales with query complexity — exactly the queries that production Presto handles.

The Fix

presto-0001/0002/0003: Snapshot `getOutputVariables()` to `ImmutableSet` before the loop:

```
// Before
if (joinNode.getLeft().getOutputVariables().contains(baseVariable)) { ... }

// After
// CWE-407 fix: snapshot to ImmutableSet before loop for O(1) contains().
Set<VariableReferenceExpression> leftOutputSet =
    ImmutableSet.copyOf(joinNode.getLeft().getOutputVariables());
if (leftOutputSet.contains(baseVariable)) { ... }
```

presto-0004: Snapshot before stream filter:

```
// Before
.filter(key -> rightNode.getOutputVariables().contains(key))

// After
// CWE-407 fix: snapshot to ImmutableSet before stream for O(1) contains().
Set<VariableReferenceExpression> rightOutputSet =
    ImmutableSet.copyOf(rightNode.getOutputVariables());
.filter(key -> rightOutputSet.contains(key))
```

`VariableReferenceExpression` implements `equals()`/`hashCode()` — no additional changes needed.

Patch

Fix available: `defects/presto/patch/presto-0001-0004-pushdown-derefs-immutableset.patch`

Two-file patch: `PushDownDereferences.java` (3 hunks) + `PayloadJoinOptimizer.java` (1 hunk).

Unit test: 5/5 pass. At D=V=100: defective=10,000 comparisons, fixed=100, **100× speedup**.

What We Ask

A patch is ready for review.

1. Confirm receipt and assign a GitHub issue or security advisory reference.
2. Assess severity — presto-0001/0002/0003 fire on every query with dereference pushdown.
3. Coordinate a disclosure date — we are targeting 90 days from first contact.
4. We will credit the Presto team in the public disclosure. Preferred acknowledgment format welcome.

Contact: see cover email. This brief is confidential until coordinated disclosure.