0-DAY NO CVE HIGH PUBLIC

SQLite rtree last_insert_rowid 0 use-after-free

SQLite 3.53.4 rtree hashes a new node as iNode 0 when last_insert_rowid stays 0. nodeHashDelete skips 0. A later split frees the RtreeNode and nodeHashLookup still finds it. ASan UAF. Not RCE. No CVE yet.

NAME SQLite rtree last_insert_rowid 0 use-after-free	TYPE 0-day analysis
CVE n/a	CVE RISK high
DISCLOSURE STATUS public	VENDOR SQLite (Hwaci)
AFFECTED SQLite 3.53.4 (b09c88c) rtree/geopoly. branch-3.53 11ce330557 is after this GA. Typical extra SQLITE_ENABLE_RTREE.	PUBLISHED 09 Oct 2026
UPDATED 09 Oct 2026	TAGS 0-day, sqlite, rtree, uaf, cwe-416

TL;DR

Lab files and proof-of-concept here.

01 Node zero is a hash key and a skip

I am @abraxas_null. The proof of concept is on GitHub: **abraxas/sqlite-rtree-rowid0** (loopback client). The lab stack is `lab/` : Dockerfile, docker-compose.yml, run.sh, poc.py, harness.c. Authorized lab only. It talks to loopback.

This is **SQLite 3.53.4** (Git `b09c88c14082339b66c7b7158d609a771e64ca69`, SOURCE_ID `2026-07-24 19:02:57 bf7c7f30031888f4e796e429ab3978879485813aaca6f641c7b33e4e09459bcc`). No CVE yet. CWE-416. 8.1 High (`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`). Local UAF. Typical extra `-DSQLITE_ENABLE_RTREE`. Not demonstrated RCE.

rtree stores nodes in a shadow table named `%_node`. After an INSERT that reports OK, `nodeWrite` asks `sqlite3_last_insert_rowid()` for the new node number and hashes that `RtreeNode`. If the rowid stays 0, the hash key is 0. `nodeHashDelete` refuses to unlink 0. `SplitNode` later `nodeRelease`s the sibling. `nodeHashLookup` still finds the freed object. AddressSanitizer calls that a heap-use-after-free READ. The process dies. That is the leftover.

Geopoly shares `nodeWrite`. Same hash. Same skip.

02 Why SQLite after OpenLiteSpeed

I had just finished unpublished Crit/High hunts of Caddy, lighttpd, and OpenLiteSpeed. Web servers argue about headers. SQLite argues about files. sqlite.org/security.html is unusually honest: untrusted SQL and untrusted `.db` (plus a sibling `-wal`) are first-class inputs. Memory corruption on those inputs is a serious bug. Default `load_extension` is off. Typical amalgamation extras are FTS5, builtin JSON, rtree/geopoly. The CLI, `sqlite3_rsync`, zipfile, session, and official WASM are extra surfaces.

I only wanted Critical or High. Local is in-scope. Medium stays on the floor unless it chains. This post is **G10 of fourteen leftovers** on the pin. The siblings are other

notes. This one is rtree node 0.

03 Pinning 3.53.4, not the draft

Latest GA, not a draft. sqlite.org/download.html listed 3.53.4 (2026-07-24). 3.54.0 was a draft with a planned date of 2026-10-15. Drafts wait.

Canonical tree is sqlite.org/src. GitHub sqlite/sqlite is a Fossil mirror. Official source zip is `sqlite-src-3530400`. Public domain. Git tag `version-3.53.4` is `b09c88c14082339b66c7b7158d609a771e64ca69`. The lab checks `sqlite_version()` and `sqlite_source_id()` before it inserts.

Hwaci writes CVEs rarely. Developers fix in hours. The leftover class on this tree is later trunk, not this GA.

04 The skip list is half the hunt

Before you look for leftovers, you write down what is already public so you do not "discover" a CVE with a press release. SQLite's table lives at sqlite.org/cves.html. On 3.53.4 the interesting ones were already closed:

- CVE-2026-11822 / CVE-2026-11824. FTS5 heap overflow when `SQLITE_DBCONFIG_DEFENSIVE` is off. `fts5LeafRead` `szLeaf<4`. Fixed 3.53.2. In tree.
- CVE-2026-50812. Session `changeset_apply_v3` NPD. Medium. Optional `-DSQLITE_ENABLE_SESSION`.
- concat_ws, LOOKASIDE, zipfile CLI, JSON UAF. Years or minors before this pin.

- Ada Logics `sqlite3_rsync` `readPow2` stack smash (`x>16`). Closed on this pin.

What is not closed on this GA: later-trunk leftovers. WAL page-size mismatch. FTS5 secure-delete footer. FTS3 `snippet()` UAF. COLLATE height. CTE nest. This post is the rtree hash.

05 Twenty hunts, one zero

I split the tree. Parser / VDBE. btree / pager. WAL. FTS3. FTS5. JSON. vtab / ATTACH / pragma. trusted_schema / authorizer. shell / zipfile. sqlite3_rsync. session / RBU. deserialize / memdb. triggers / views / window. rtree / geopoly. WASM. 3.54 patch siblings. load_extension / VFS. C API. recover / misc extras. schema side effects.

Most of that is SQLite doing what 3.53.4 said it would do.

`load_extension` stays off. JSON UAF class is closed. Default `TRUSTED_SCHEMA` is ON. rtree sets `SQLITE_VTAB_INNOCUOUS`, so it still runs with trusted_schema off. Default compile does not set `SQLITE_DEFAULT_DEFENSIVE`. A pre-planted trigger in an untrusted DB file still runs when rtree's own `pWriteNode` INSERT executes.

Official recommended compile, Windows nmake `OPTS`, autoconf fallback, and fuzzcheck all set `-DSQLITE_ENABLE_RTREE`. If you built the docs, you have the virtual table.

06 Then I opened nodeWrite

C Copy

```c
if( pNode->iNode==0 && rc==SQLITE_OK ){
  pNode->iNode = sqlite3_last_insert_rowid(pRtree->db);

```

```
  nodeHashInsert(pRtree, pNode);
}
```

That is the assignment. If the INSERT into `%_node` reports OK and the connection's last rowid is still 0, the hash key is 0.

C Copy

```c
static void nodeHashDelete(Rtree *pRtree, RtreeNode *pNode){
  RtreeNode **pp;
  if( pNode->iNode!=0 ){
    pp = &pRtree->aHash[nodeHash(pNode->iNode)];
    for( ; (*pp)!=pNode; pp = &(*pp)->pNext){ assert(*pp); }
    *pp = pNode->pNext;
    pNode->pNext = 0;
  }
}
```

Zero is a skip. `nodeRelease` after `SplitNode` frees the object. `nodeAcquire` looks up the hash before it rejects `iNode ⩽ 0`. A hit on the freed zero entry is the UAF. A miss on `iNode ⩽ 0` is `SQLITE_CORRUPT_VTAB`, which is a different path.

Documented ways to keep last_insert_rowid at 0: replace `%_node` with WITHOUT ROWID, or plant `BEFORE INSERT RAISE(IGNORE)` on the shadow table and reopen so the connection's last rowid is 0. Then INSERT until a node split. The right sibling is hashed as node 0, released, and still findable.

07 The lab

Host is macOS. One container. Compose project `sqlite-rtree-rowid0`. Loopback `127.0.0.1:18890`. ASan amalgamation of official `sqlite-src-3530400` with `-DSQLITE_ENABLE_RTREE=1`. Harness is `lab/harness.c`. Bind loopback.

The sequence is vendor `rtreeI.test` 2.0-2.2: fill 51 rows (`RTREE_MAXCELLS`), plant `BEFORE INSERT RAISE(IGNORE)` on `%_node`, reopen so `last_insert_rowid()==0`,

INSERT the 52nd row (split), then `SELECT *`. Control split without the trigger is OK.

Wrong turns, in order: treating `%_rowid` last_insert_rowid 0 as this bug (that assigns cell rowid 0, not the hash UAF); expecting `SQLITE_CORRUPT_VTAB` on this pin (that is the later fix); filling 50 cells so there is no split; planting the trigger and not reopening; lookaside hiding the free from ASan; calling the crash RCE.

Last run:

PLAIN TEXT Copy

```
control-split=ok
reopen-last-rowid=0
witness-before-split=SQLITE-RTREE-ROWID0-WITNESS
ERROR: AddressSanitizer: heap-use-after-free
READ of size 8 ... nodeHashLookup
freed by ... nodeRelease / SplitNode
SUCCESS sqlite-rtree-rowid0 how=asan-uaf-nodehash SQLITE-RTREE-ROWID0-WITNESS
```

Replay: clone the pack, `cd lab && ./run.sh`.

08 What an attacker can do

On an rtree or geopoly table, make node INSERT report OK while `last_insert_rowid()` stays 0, then INSERT until a split. The host process uses a freed `RtreeNode`. Lab is AddressSanitizer heap-use-after-free. Instruction-pointer control is not in this pack. The crash is.

09 The fix

After `sqlite3_last_insert_rowid()`, if `iNode` is still 0, return `SQLITE_CORRUPT_VTAB` and do not `nodeHashInsert`. branch-3.53 `11ce330557` / Git

`63b6d705` (2026-09-24, Claude / Ada Logics) does that. The test expects "database disk image is malformed". This GA does not.

10 References

- github.com/abraxas/sqlite-rtree-rowid0
- sqlite.org/src tag version-3.53.4 (`b09c88c14082339b66c7b7158d609a771e64ca69`)
- `ext/rtree/rtree.c` `nodeWrite`, `nodeHashDelete`, `nodeRelease`, `nodeAcquire`, `SplitNode`
- branch-3.53 fix `11ce330557` (2026-09-24)
- CWE-416

- Contact: abraxaslabs.tech · github.com/abraxas · @abraxas_null · abraxas.null@proton.me

Abraxas Labs

Independent security research by abraxas.

Analysis of 0-days, n-days, malware, and the

leftovers that still matter.

GitHub **@abraxas_null** **abraxas.null@proton.me** **abraxaslabs.tech**

Independent research. Private analysis by arrangement. No ads. abraxas.null@proton.me · For defensive awareness and authorized testing only.

© 2026 Abraxas Labs