0-DAY NO CVE HIGH PUBLIC

# SQLite FTS5 detail=none prefix-merge under-alloc

SQLite 3.53.4 FTS5 detail=none prefix merge sizes the output as p1→n + p2→n, then fts5BufferSafeAppendVarint writes past that allocation. Twin fts5MergePrefixLists already budgets + 9. Labbed ASan WRITE of size 1. Crash. No CVE yet.

| | |
|---|---|
| NAME<br>SQLite FTS5 detail=none prefix-merge under-alloc | TYPE<br>0-day analysis |
| CVE<br>n/a | CVE RISK<br>high |
| DISCLOSURE STATUS<br>public | VENDOR<br>SQLite (Hwaci) |
| AFFECTED<br>SQLite 3.53.4 (b09c88c) FTS5 with detail=none. SOURCE_ID 2026-07-24 19:02:57 bf7c7f30031888f4e796e429ab3978879485813aaca6f641c7b33e4e09459bcc. Default detail=full is not this leftover. Trunk a7f8c92f62 is after this GA. | PUBLISHED<br>09 Oct 2026 |
| UPDATED<br>09 Oct 2026 | TAGS<br>0-day, sqlite, fts5, cwe-787, crash, heap-overflow |

TL;DR

Lab files and proof-of-concept here.

## 01 The twin already knew about the extra varint

I am @abraxas_null. The proof of concept is on GitHub: **abraxas/sqlite-fts5-rowidmerge** (loopback client). The lab stack is `lab/` : Dockerfile, docker-compose.yml, run.sh, poc.py, harness.py. Authorized lab only. It talks to loopback.

This is **SQLite 3.53.4** ( `b09c88c14082339b66c7b7158d609a771e64ca69` ). Fossil SOURCE_ID `2026-07-24 19:02:57 bf7c7f30031888f4e796e429ab3978879485813aaca6f641c7b33e4e09459bcc` . No CVE yet. CWE-787. 7.5 High ( `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H` ). Local crash. FTS5 `detail=none` . Prefix MATCH. ASan WRITE of size 1. Not demonstrated RCE.

FTS5 stores inverted lists as delta-encoded varints. Merge two lists and a rowid's predecessor can change. A large negative first rowid makes the next delta wider than it was in its own list. `fts5MergePrefixLists` documents that case and budgets `p1→n + 9` . `fts5MergeRowidLists` , the `detail=none` twin, sizes with `p1→n + p2→n` and then calls `fts5BufferSafeAppendVarint` , which does not grow. The comment and the allocator live a hundred lines apart. One of them got the slack.

Dan Kennedy fixed it on trunk the same morning Michele posted the report. Latest GA is still 3.53.4. 3.54.0 is a draft. The leftover is the release that did not pick up the `+ 9` .

## 02 Why SQLite

I had just finished unpublished Crit/High hunts of Caddy, lighttpd, OpenLiteSpeed, and MySQL Community 26.7.0. Caddy and lighttpd Unix came back empty at this bar. OpenLiteSpeed's leftover was an HTTP/1 duplicate Cookie that skipped a 403. MySQL's leftovers were client tools and authenticated SQL, not default unauth `mysqld` . The next tree that actually sits under other people's processes is SQLite.

SQLite is not a daemon. It is a library that becomes whatever hosts it. Browsers, phones, language bindings, "we store the session in a file" web apps. sqlite.org/security.html is unusually honest about the input: untrusted SQL and untrusted database files (plus a sibling `-wal` ) are first-class. Memory corruption on those inputs is High. `load_extension` is off on a default open. `TRUSTED_SCHEMA` is on. Typical amalgamation extras are FTS5, JSON, rtree. That is the product people compile.

I only wanted Critical or High. Local is in-scope. Authenticated SQL DoS is below the bar unless it is memory corruption. `INTO OUTFILE` does not exist here. Superuser

`load_extension` is intended and off by default. WASM, the CLI, and `sqlite3_rsync` are extra surfaces. I am not filing "ATTACH a file."

## 03 Pinning 3.53.4

Canonical source is sqlite.org/src. GitHub sqlite/sqlite is a Fossil mirror. People clone the mirror and then argue about tag objects. The useful pin is Fossil SOURCE_ID plus the Git commit the mirror attached to the `version-3.53.4` tag.

3.53.4 is 2026-07-24. Git `b09c88c14082339b66c7b7158d609a771e64ca69`. SOURCE_ID `2026-07-24 19:02:57 bf7c7f30031888f4e796e429ab3978879485813aaca6f641c7b33e4e09459bcc`. Download zip `sqlite-src-3530400.zip`, SHA3-256 `b834d474b9b393d85a9e3ee4cc11f1329e007e9376a424ee740796f5c4bda3a8`. The lab hashes that zip before it compiles anything.

3.54.0 was on the calendar for 2026-10-15 and still a draft when I pinned. Drafts move. GA does not, until it does. I did not pin a draft.

3.53.4 itself is a maintenance wave for (mostly) AI-discovered bugs in 3.53.0 through 3.53.3. If a leftover of later trunk still fires here, it is not a stale minor. It is the release people ship.

## 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 list 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. Duplicate CVEs for the same `szLeaf<4` underflow. Fixed 3.53.2. This pin's `fts5LeafRead` rejects `szLeaf<4` or `szLeaf>nn`.
- CVE-2026-50812 session `sqlite3changeset_apply_v3` NPD. Medium. Optional `-DSQLITE_ENABLE_SESSION`.
- CVE-2025-3277 / CVE-2025-29087 `concat_ws` integer overflow. Fixed 3.49.1.

- CVE-2025-6965 integer overflow read. Fixed 3.50.2.
- CVE-2025-7709 FTS5 corrupt-index overflow. Fixed 3.50.3.
- CVE-2025-29088 LOOKASIDE OOB. Fixed 3.49.1.
- CVE-2025-70873 zipfile CLI. Fixed 3.52.0. Extra, not default lib.
- CVE-2024-0232 JSON UAF. Fixed 3.43.2.
- Ada Logics `sqlite3_rsync` `readPow2` stack smash: already in this pin ( `x>16` ).
- WAL-reset corruption (Tailscale race): 3.51.3 / 3.53.0.

The 2026-06/07 FTS5 corrupt-record over-read wave is in `fts5corruptB.test` on this GA. I am not re-filing `szLeaf`. I am not re-filing JFrog's unreproducible CVE cluster. I am not filing experimental `.expert`.

Public forum leftovers of later trunk versus this pin are a different pile. WAL page-size mismatch. FTS5 secure-delete footer. This merge. rtree `last_insert_rowid` 0. Authorizer FULL JOIN USING. `sqlite3_rsync` SSH filenames. They are on sqlite.org/bugs with no CVE and no 3.53.4 release. This post is one of them.

## 05 Twenty surfaces, one merge that still adds wrong

I split the tree. Parser and VDBE. B-tree and pager. WAL. FTS3. FTS5. JSON. vtab / ATTACH / pragma. `trusted_schema` and authorizer. Shell and zipfile. `sqlite3_rsync`. Session and RBU. deserialize / memdb. Triggers, views, window. rtree / geopoly. WASM. 3.54 patch siblings. `load_extension` and VFS. C API. recover extras. Schema side effects.

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

`fts5LeafRead` fail-closes the 11824 underflow. JSON is not the 2024 UAF. `concat_ws` is i64 plus `malloc64`. Default open does not load extensions. Integrity-check over-reads that only fire on `INSERT INTO t(t) VALUES('integrity-check')` were already patched on this GA. Porter signed shift is UBSAN, not a heap write. `fts5DlidxIterInit` uncapped height is a stack crash from planted shadow tables, smaller write surface, not this leftover.

FTS5 is a typical extra. `SQLITE_ENABLE_FTS5` is on in amalgamation fuzz flags and in anyone who shipped full-text search. Default `CREATE VIRTUAL TABLE t USING fts5(a)` is `detail=full`. `detail=none` is a documented size knob: omit positions and column hits, store rowids only. Apps that want a smaller index turn it on. The merge function for that mode is `fts5MergeRowidLists`.

## 06 detail=none, prefix MATCH, no prefix= required

`fts5SetupPrefixIter` picks the merger from the table option:

```c
if( p->pConfig->eDetail==FTS5_DETAIL_NONE ){
  s.xMerge = fts5MergeRowidLists;
  s.xAppend = fts5AppendRowid;
}else{
  s.nMerge = FTS5_MERGE_NLIST-1;
  s.nBuf = s.nMerge*8;
  s.xMerge = fts5MergePrefixLists;
  s.xAppend = fts5AppendPoslist;
}
```

`ext/fts5/fts5_index.c:6727-6735` on this pin. Prefix query, `detail=none`, rowid lists. You do not need the `prefix=` create option. `MATCH 'a*'` still walks main terms that start with `a` and merges their doclists. `ab` and `abc` are two terms. Two lists. One merge.

The prefix-list merger, used when detail is not none, already writes the comment this leftover is:

```c
  /* The maximum size of the output is equal to the sum of the
  ** input sizes + 1 varint (9 bytes). The extra varint is because if the
  ** first rowid in one input is a large negative number, and the first in
  ** the other a non-negative number, the delta for the non-negative
  ** number will be larger on disk than the literal integer value
  ** was.
```

`fts5_index.c:6207-6211`. Then it allocates `p1->n + 9 + 10*nBuf`. The `detail=none` twin is "the equivalent of `fts5MergePrefixLists()`" and then does not take the equivalent

budget:

```c
static void fts5MergeRowidLists(
  Fts5Index *p,
  Fts5Buffer *p1,
  int nBuf,
  Fts5Buffer *aBuf
){
  ...
  sqlite3Fts5BufferSize(&p->rc, &out, p1->n + p2->n);
  if( p->rc ) return;

  fts5NextRowid(p1, &i1, &iRowid1);
  fts5NextRowid(p2, &i2, &iRowid2);
  while( i1>=0 || i2>=0 ){
    if( i1>=0 && (i2<0 || iRowid1<iRowid2) ){
      fts5BufferSafeAppendVarint(&out, iRowid1 - iOut);
      ...
    }else{
      fts5BufferSafeAppendVarint(&out, iRowid2 - iOut);
      ...
    }
  }
```

`fts5_index.c:6090-6127`. `fts5BufferSafeAppendVarint` is a macro. It puts a varint at `p[n]` and adds the length. It does not realloc. The assert is `nSpace>=n` after the write, which is how you find this under ASan and miss it in a release build that does not trip the next chunk until later.

`sqlite3Fts5BufferSize` in `fts5_buffer.c:18-24` starts at 64 and doubles. Ask for 64, get 64. Ask for 65, get 128. The interesting input is one whose encoded sizes sum to 64.

## 07 Why 55, and why INT64_MIN

Rowids 1 through 55, all term `ab`, delta-encode as fifty-five one-byte varints. That list is 55 bytes. Term `abc` at rowid `-9223372036854775808` (`INT64_MIN`, `SMALLEST_INT64`) is a nine-byte varint. Sum 64. Allocator returns 64.

`MATCH 'a*'` merges `ab` and `abc`. `INT64_MIN` is first in the output (smallest rowid, keeps its nine-byte encoding). The next rowid is 1. The delta is `1 - INT64_MIN`, a nine-byte

unsigned varint, where the original `ab` list encoded that 1 in one byte. Then fifty-four more one-byte deltas. Output wants 9 + 9 + 54 = 72. Allocation is 64. Eight bytes past the end, the first of them a WRITE of size 1 inside `sqlite3Fts5PutVarint`.

That is not an accident of 55. `sqlite3Fts5BufferSize` rounds to a power of two. The reproducer has to land the input sum on that boundary or the overflow is absorbed by slack you did not earn. Michele's forum post says this in one paragraph. I counted it again against 3.53.4 anyway, because forum SQL against a draft SOURCE_ID is not a lab on the GA zip.

Control is the same table, `MATCH 'ab'`. One term. No merge of a negative head into a positive list. 55 rows. `rc=0`. No ASan.

No corrupt file. No `SQLITE_DBCONFIG_DEFENSIVE` off. No shadow-table SQL. Create the table the documented way, insert ordinary rows plus one negative rowid, prefix MATCH.

## 08 The image is ASan sqlite3, not in-process

ASan `abort_on_error` in the same process as the HTTP harness is a lab that dies before it can tell you it died. The Dockerfile builds `sqlite-src-3530400` with `-O1 -g -fsanitize=address -DSQLITE_ENABLE_FTS5`, installs `/usr/local/bin/sqlite3`, and runs `harness.py` as a loopback SQL runner. `POST /sql` execs the CLI on `:memory:` with a timeout. stderr comes back as JSON. The PoC lives on the host and talks to `127.0.0.1:18870`.

FTS5 is off unless `--enable-fts5`. The health check refuses to proceed unless `sqlite_version` is `3.53.4`, SOURCE_ID contains `2026-07-24`, and `CREATE VIRTUAL TABLE t USING fts5(a)` works. If you skip that you will lab a distro sqlite3 and write a FAIL about the distro.

Compose project `sqlite-fts5-rowidmerge`. Bind `127.0.0.1:18870:8080`. `ASAN_OPTIONS=detect_leaks=0:abort_on_error=1:halt_on_error=1:allocator_may_return_null=1`.

## 09 The lab

Control `MATCH 'ab'`: HTTP 200, `rc=0`, stdout `55`, no heap-buffer-overflow, no `fts5MergeRowidLists` frame.

Trigger `MATCH 'a*'`: ASan `heap-buffer-overflow`, `WRITE of size 1`, frames `sqlite3Fts5PutVarint` then `fts5MergeRowidLists`. `rc=-6`. No row count.

Last line:

```
PLAIN TEXT
Copy
1 SUCCESS sqlite-fts5-rowidmerge how=asan-heap-buffer-overflow-fts5MergeRowidLists version=3.5
```

Replay:

```
PLAIN TEXT
Copy
1 cd lab && ./run.sh
```

Or from a fresh clone:

```
PLAIN TEXT
Copy
1 git clone https://github.com/abraxas/sqlite-fts5-rowidmerge
2 cd sqlite-fts5-rowidmerge/lab
3 ./run.sh
```

Bind it to loopback. The loopback client is `lab/run.sh` with a banner on it. I am not going to paste a heap-grooming payload. The oracle is the ASan WRITE and the sink frame.

## 10 Wrong turns, in the order I made them

**Pinning 3.54.0.** It looks like "latest." It is a draft. Michele's report even names a 3.54.0 SOURCE_ID. The leftover that matters for operators is the GA zip they actually install. 3.53.4.

**Treating GitHub as canonical.** `sqlite/sqlite` is a mirror. Tags, SOURCE_ID, and the download SHA3 live on sqlite.org. The lab fetches `sqlite-src-3530400.zip` and checks the hash. If GitHub and Fossil ever disagree, Fossil wins.

**Re-finding CVE-2026-11824.** `fts5LeafRead` already rejects `szLeaf<4`. Continuation pages go through that helper. The 2026-06/07 corrupt-leaf wave is closed. This leftover is a merge of well-formed doclists, not a short leaf.

**Requiring `prefix=`.** Prefix indexes are an optimization. `MATCH 'a*'` against the main terms still merges. The lab does not set `prefix=`. If you add it you are testing a different iterator.

**Using default `detail=full`.** Then `xMerge` is `fts5MergePrefixLists`, which already has the `+ 9`. The control is not "prefix MATCH is safe." The control is "one term, no negative-head merge."

**Running ASan sqlite3 inside the HTTP process.** First crash kills the harness. You get a compose restart loop and a FAIL that is about PID 1, not `fts5MergeRowidLists`. Subprocess the CLI. Capture stderr.

**Calling a 1-byte heap write RCE.** It is a WRITE of size 1 in a varint encoder. With a hostile allocator and a week you can tell a story. The lab does not. Class is Crash. sqlite.org's own model still scores untrusted-SQL memory corruption as High. That is why the vector is `AV:N` on a local library.

**Confusing this with the FTS5 secure-delete footer overrun.** Different function (`fts5DoSecureDelete`), different trigger (DELETE/flush on a table already in secure-delete, usually a crafted `%_data` leaf). Same file, same `sqlite3Fts5PutVarint`, not this leftover. Trunk fixed that one eleven days later as YEONBA. This one is Michele, 2026-08-04, `a7f8c92f62`.

**Expecting `MATCH 'a*'` to need a planted database file.** It does not. Untrusted SQL against a `detail=none` table is enough. Untrusted `.db` that already has that table is the other shape of the same input.

**Skipping FTS5 because "it is an extension."** It is the full-text search people enable. The amalgamation extras are where the leftover hunts live once core parser CVEs are closed.

## 11 What an attacker can do

On an FTS5 table created with `detail=none`, run a prefix MATCH that merges a rowid list headed by a large negative rowid into a list of ordinary rowids. `fts5MergeRowidLists` under-allocates. `fts5BufferSafeAppendVarint` writes past the buffer. The host process that opened that database crashes. Under ASan the crash is a heap-buffer-overflow WRITE of size 1 in `sqlite3Fts5PutVarint`.

Victim is the embedder: a CLI, a language binding, a web worker that takes SQL or opens a file. SQLite's security docs already treat those as attack surface. This is not remote

unauthenticated RCE in `libsqlite3`. It is not a demonstrated write-what-where. It is a High crash on a documented table option and a documented query syntax.

Default `detail=full` tables are not this function. Tables without a prefix query do not merge these lists. `MATCH 'ab'` on the lab table returns 55 rows and keeps the process.

## 12 The fix

Budget the `detail=none` merge the way `fts5MergePrefixLists` already does:

```c
sqlite3Fts5BufferSize(&p->rc, &out, p1->n + p2->n + 9);
```

That is a7f8c92f62 / branch-3.53 `f8c72d27b2`. One extra varint is enough: at most one delta in the output grows, from at least 1 byte to at most 9. Cast the delta to `u64` so `INT64_MIN` does not signed-overflow on the way in, matching `fts5MergeAppendDocid`. Or stop using the Safe append and grow on demand.

Until 3.53.x picks that up, do not put untrusted SQL against FTS5 `detail=none` in a process you care about keeping. Do not open untrusted `detail=none` databases in that process either. Turning FTS5 off is a compile option. Changing `detail=none` to `detail=full` is a create-time option. Neither is a patch.

## 13 References

- Proof of concept: abraxas/sqlite-fts5-rowidmerge · sqlite-fts5-rowidmerge-Abraxas-Labs.py
- Lab: `lab/` · Dockerfile · docker-compose.yml · run.sh · poc.py · harness.py
- @abraxas_null · github.com/abraxas · abraxaslabs.tech · abraxas.null@proton.me
- CWE-787 · CVSS 7.5

- Tree: sqlite.org/src tag version-3.53.4 ( `b09c88c14082339b66c7b7158d609a771e64ca69` ) · SOURCE_ID `2026-07-24 19:02:57 bf7c7f30031888f4e796e429ab3978879485813aaca6f641c7b33e4e09459bcc`
- Sink: `fts5_index.c` `fts5MergeRowidLists` · twin comment in `fts5MergePrefixLists` · `fts5_buffer.c` `sqlite3Fts5BufferSize`
- Forum: 2026-08-04T08:49:27Z (Michele) · trunk fix a7f8c92f62
- FTS5 detail option · sqlite.org/security.html · sqlite.org/cves.html

**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