0-DAY NO CVE HIGH PUBLIC

SQLite 3.53.4 FTS3 snippet() use-after-free

SQLite 3.53.4 FTS3/4 snippet() still walks poslist pointers after fts3EvalRestart frees the phrase doclist. ASan heap-use-after-free READ in fts3SnippetAdvance. Local UAF. Not RCE. No CVE yet.

NAME SQLite 3.53.4 FTS3 snippet() 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) FTS3/4. Trunk e4635c4153 / branch-3.53 9a367c177f are after this GA. CLI OPT_SHELL enables FTS4.	PUBLISHED 09 Oct 2026
UPDATED 09 Oct 2026	TAGS 0-day, sqlite, fts3, snippet, uaf, cwe-416

TL;DR

Lab files and proof-of-concept here.

01 snippet() is not supposed to free the buffer it still walks

I am @abraxas_null. The proof of concept is on GitHub: **abraxas/sqlite-fts3-snippet** (loopback client). The lab stack is `lab/`: Dockerfile, docker-compose.yml, run.sh, poc.py, health.py. Authorized lab only. It talks to loopback.

This is **SQLite 3.53.4** (`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. Class is **UAF**. Not demonstrated RCE.

`snippet()` is the helper that turns a MATCH into a highlighted fragment. Search UIs call it because nobody wants to ship the whole document back just to bold three words. On 3.53.4, a MATCH that mixes an incremental phrase with OR and NEAR makes `fts3EvalRestart` free a phrase doclist while `fts3BestSnippet` still holds pointers into that allocation. The next step is `fts3SnippetAdvance`. AddressSanitizer calls that a heap-use-after-free READ. The process dies. That is the leftover.

Control MATCH `hello` plus `snippet()` is fine. The trigger is the vendor test string, not a payload.

02 Why SQLite

I had just finished unpublished Crit/High hunts of HTTP servers and SQL clients. Default daemons came back empty or leftover-of-a-named-guard. SQLite is a different animal. It is public domain. It is in phones, browsers, edge boxes, and every language binding that got tired of shipping a server. sqlite.org/security.html is honest about the victim model: untrusted SQL and untrusted `.db` files (plus a sibling `-wal`) are first-class inputs. Memory corruption on those inputs is a security bug. `load_extension` is off on default open. That is not a reason to stop reading `ext/fts3/`.

I only wanted Critical or High. Local is in-scope. Medium stays on the floor unless it chains. A finding on this hunt needed leftover memory safety of a later trunk fix, a crash the documented limits were supposed to stop, or an extra that ships in the CLI / recommended compile options and still frees a buffer the caller walks.

The interesting part of SQLite is not that it has bugs. The interesting part is that the developers fix in hours, often without a CVE, and the latest GA can still be sitting behind the check-in by a month. That is a leftover class. Hunt the GA. Do not pin the draft.

03 Pinning 3.53.4

Latest GA. Not an RC. Not 3.54.0. Downloads and the Fossil timeline both said 3.53.4 (2026-07-24). 3.54.0 was still a draft with a planned 2026-10-15. Drafts wait.

GitHub tag `version-3.53.4` is a Fossil mirror. Canonical is sqlite.org/src. HEAD landed on `b09c88c14082339b66c7b7158d609a771e64ca69`. `SELECT sqlite_source_id();` on the amalgamation is `2026-07-24 19:02:57 bf7c7f30031888f4e796e429ab3978879485813aaca6f641c7b33e4e09459bcc`. The official zip is `sqlite-src-3530400.zip`. That is the pin.

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

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. Fixed 3.53.2. `fts5LeafRead` `szLeaf<4` is in this pin. Not this leftover.
- CVE-2026-50812 session `sqlite3changeset_apply_v3` NPD. Medium. Optional `-DSQLITE_ENABLE_SESSION`.
- concat_ws integer overflow / LOOKASIDE OOB / FTS5 corrupt-index integer overflow / zipfile CLI OOB. 3.49-3.52.
- CVE-2024-0232 JSON UAF. Closed.
- Magellan-era FTS3 shadow-table integer overflows (CVE-2018-20346, CVE-2018-20506, 2020 FTS3). Years before this pin.
- Fossil `85afd8e291` (2026-06-12): NEAR buffer overwrite on corrupt records. `fts3PutDeltaVarint` refuses a negative delta. In this pin.
- Fossil `42fe4d2e8b` (2026-06-12): signed overflow in `fts3ReadEndBlockField`. `SMALLEST_INT64` guard is in this pin.
- `adcb44f9dc` NEAR/N distance clamp, `c0f2cd6d29` matchinfo `'b'` bitmap, `d903795e3b` / `ef0c66d2a2` fts3aux, `df5728d79f` deferred-phrase padding, `b2816b4d6a` `fts3StringAppend` grow. All in tree.
- Ada Logics `sqlite3_rsync` stack smash (`readPow2` `x>16`). Closed on this pin. 3.53.4 is 2026-07-24; that fix landed earlier.
- JFrog "critical CVEs" 51296/51297/51300... unreproducible. Skip.

Historical `SQLITE_ENABLE_FTS3_TOKENIZER` is still a non-default extra. I am not re-filing a compile flag nobody ships.

I also left on the floor: FTS3 integrity-check integer truncation (checksum bypass, Dan already said those checks are not cryptographic), a small GetVarint overread on the integrity path, `snippet()` 64-bit argument truncation, and the out-of-range snippet column that falls back to column 0. Low. Not this post.

05 Twenty surfaces, one FTS3 leftover

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. `sqlite_log()` from a hostile schema is log injection, Medium, and 3.54 DIRECTONLY was not backported. I am not filing a log line.

FTS5 on this pin already ate `szLeaf<4`. The FTS5 leftovers that survived are a different extra (`secure-delete` , `detail=none` merge) and a different post. rtree `last_insert_rowid()==0` is a different extra and a different post. This one is FTS3/4 `snippet()` .

FTS3/4 is a typical extra. The library amalgamation does not enable it unless you pass `-DSQLITE_ENABLE_FTS3` or `SQLITE_ENABLE_FTS4` . The CLI default `OPT_SHELL` enables FTS4. Debian, Android, and the recommended compile options often ship it. If you built `sqlite3` with the docs, you have the virtual table. Untrusted MATCH plus `snippet()` is then a local input. Untrusted FTS shadow tables are the same victim model as every other FTS leftover since Magellan.

06 Then I opened snippet()

`fts3BestSnippet` decides which fragment to highlight. It counts phrases, allocates a `SnippetIter`, and asks `sqlite3Fts3ExprIterate` to fill `aPhrase[]` with poslist pointers:

```c
rc = sqlite3Fts3ExprIterate(
    pCsr->pExpr, fts3SnippetFindPositions, (void*)&sIter
);
```

`fts3SnippetFindPositions` calls `sqlite3Fts3EvalPhrasePoslist`, then caches `pHead` / `pTail` into that array. After the iterate returns, `fts3SnippetNextCandidate` / `fts3SnippetAdvance` walk those pointers as if they still belonged to live doclists.

For a later phrase under OR, `sqlite3Fts3EvalPhrasePoslist` sees `pPhrase->bIncr` and cannot keep using an incremental phrase:

```c
if( bOr==0 ) return SQLITE_OK;
/* ... */
if( pPhrase->bIncr ){
  int bEofSave = pRun->bEof;
  fts3EvalRestart(pCsr, pRun, &rc);
```

`fts3EvalRestart` walks the subtree and invalidates every phrase poslist it finds:

```c
static void fts3EvalInvalidatePoslist(Fts3Phrase *pPhrase){
  if( pPhrase->doclist.bFreeList ){
    sqlite3_free(pPhrase->doclist.pList);
  }
  pPhrase->doclist.pList = 0;
```

The pointers cached for the earlier phrases now dangle. `fts3SnippetAdvance` dereferences them:

```c
static void fts3SnippetAdvance(char **ppIter, i64 *piIter, int iNext){
  char *pIter = *ppIter;
  if( pIter ){
    i64 iIter = *piIter;
    while( iIter<iNext ){
      if( 0==(*pIter & 0xFE) ){
```

That load is the ASan READ. The free is `fts3EvalRestart`. The class is use-after-free. I am not going to dress it up as remote code execution because a heap UAF can become one on a lucky allocator. The lab is a crash under AddressSanitizer. That is what I will claim.

`offsets()` already restarts first (`fts3ExprRestartIfCb`). That is why `offsets()` is not this leftover. `matchinfo()` LCS caches `pRead` across per-phrase `sqlite3Fts3EvalPhrasePoslist` and then walks them - a sibling of the same restart class. The vendor patch only touched `fts3BestSnippet`. I labbed `snippet()`.

The MATCH that trips it is not clever. It is test 4.1 from later trunk:

```
CREATE VIRTUAL TABLE ft0 USING fts4;
INSERT INTO ft0 VALUES('0*bbb a');
SELECT snippet(ft0,'') FROM ft0 WHERE ft0 MATCH '"a" OR "" bbb NEAR 0 NEAR 1';
```

Parenthesis MATCH needs `SQLITE_ENABLE_FTS3_PARENTHESIS` (the CLI has it; the vendor repro sets `sqlite_fts3_enable_parentheses 1`). OR exists in legacy syntax too. The restart path requires an OR ancestor: `bOr==0` returns before `fts3EvalRestart`. NEAR is how you get an incremental phrase that still has to become a full doclist.

07 The forum got there first. The GA did not.

I am not going to pretend this sink was a secret. Marc Heuse reported it on 2026-08-17 against a 3.54.0 snapshot. Dan fixed trunk the same afternoon: e4635c4153, "Avoid a problem causing a use-after-free in the fts3 snippet() function." Cherry-pick to `branch-3.53` is 9a367c177f on 2026-08-18.

3.53.4 shipped 2026-07-24. The leftover is the latest GA, sitting in front of a one-line iterate that trunk replaced with two. Later trunk iterates twice "in case pointers garnered during the first iteration are invalidated by a call to fts5EvalRestart()" - the comment even says `fts5` when the function is `fts3EvalRestart`. I will take the second iterate over the comment.

No CVE. SQLite often does not write them. The skip list is CVEs plus named Fossil check-ins. A public forum thread with a trunk fix is not the same thing as "3.53.4 is patched." It is the opposite: a dated reason to lab the GA.

08 The lab

Host is macOS. One container.

Image is an ASan amalgamation of official `sqlite-src-3530400.zip`, size-pinned at 14557315 bytes, built with `-O1 -g -fsanitize=address -fno-omit-frame-pointer` and `-DSQLITE_ENABLE_FTS3=1 -DSQLITE_ENABLE_FTS3_PARENTHESIS=1 -DSQLITE_ENABLE_FTS4=1`. Runtime has `libasan8` and `llvm-symbolizer`. `ASAN_OPTIONS=detect_leaks=0:abort_on_error=1:halt_on_error=1:symbolize=1`.

Compose project `sqlite-fts3-snippet`. Published `127.0.0.1:18880` to container 8080. health.py answers `sqlite-fts3-snippet ready`. There is no network listener for the library. The port is a readiness oracle so `run.sh` does not race `sqlite3`. Bind it to loopback.

The PoC waits for health, then `SELECT sqlite_version(); SELECT sqlite_source_id();` inside the container. Pin mismatch is FAIL. Then two in-

memory sessions:

01 Control: `CREATE VIRTUAL TABLE t USING fts4; INSERT ... 'hello world'; SELECT snippet(t,'') FROM t WHERE t MATCH 'hello';`. Must return a snippet containing `hello`. Must not trip ASan. Must exit 0.

02 Trigger: the MATCH above plus `snippet(ft0,'')`. Must be `heap-use-after-free`. Frames must name `fts3SnippetAdvance` / `fts3BestSnippet` and `fts3EvalRestart`. Abort 134 is the ASan halt, not a successful SELECT.

Last run:

PLAIN TEXT Copy

```
health=200 host=127.0.0.1 port=18880
sqlite3-rc=0 id=3.53.4
2026-07-24 19:02:57 bf7c7f30031888f4e796e429ab3978879485813aaca6f641c7b33e4e0945
control-rc=0 uaf=no out=hello</b> world
trigger-rc=134
asan ==19==ERROR: AddressSanitizer: heap-use-after-free ... fts3SnippetAdvance
asan #9 ... fts3EvalRestart
asan SUMMARY: AddressSanitizer: heap-use-after-free ... fts3SnippetAdvance
asan-snippet-frame=yes
asan-restart-frame=yes
SUCCESS sqlite-fts3-snippet how=asan-heap-uaf-snippet-evalrestart version=3.53.4
```

Replay:

PLAIN TEXT Copy

```
git clone https://github.com/abraxas/sqlite-fts3-snippet
cd sqlite-fts3-snippet/lab
./run.sh
```

Bind it to loopback. The loopback client is `lab/run.sh` with a banner on it. Stack comes down after the run.

09 Wrong turns, in the order I made them

Pinning 3.54.0. It looks like "latest." It was a draft. Marc reproduced against a 3.54 snapshot dated 2026-07-31, which is after 3.53.4 and still before the 2026-08-17 fix. The leftover that matters to operators is the GA they can download without a Fossil checkout.

Treating a public forum thread as a CVE on the pin. Trunk `e4635c4153` is the fix. 3.53.4 is not that check-in. "Dan already patched it" is true for trunk and false for the release.

Re-filing CVE-2026-11824. FTS5 `szLeaf<4` is a different virtual table, a different file, and already in 3.53.2. `snippet()` is FTS3/4. I grepped `fts5` out of the ASan frames on purpose.

Re-filing Magellan. Shadow-table integer overflow is 2018-2020. The skip list exists so you do not spend a week rediscovering `fts3_write.c` history. This leftover is a poslist lifetime bug in `snippet()`, not a varint length.

Treating NEAR overwrite `85afd8e291` as still open. The negative-delta guard is in this pin. `fts3PutDeltaVarint` is not `fts3EvalRestart`. Same directory, different decade of leftover.

Calling `offsets()` the same bug. `offsets()` restarts the expression before it walks. `snippet()` caches, then walks, then (for a later phrase) restarts underneath the cache. The vendor patch is a second iterate in `fts3BestSnippet`. That is the whole fix. One function.

Expecting a default amalgamation without FTS3. `-DSQLITE_ENABLE_FTS3` is off unless you pass it. The CLI is not that amalgamation. `OPT_SHELL` enables FTS4. A

leftover that only fires with FTS compiled in is still High when that extra is how `sqlite3` ships.

Calling a heap UAF remote code execution. ASan READ of a freed poslist is UAF. The lab does not groom the heap, does not claim RIP, and does not ship a payload. Crash / memory corruption. If someone wants to turn this into an allocator story, that is a different paper with a different oracle. This pack will not be that paper.

Skipping parenthesis MATCH. The vendor test sets `sqlite_fts3_enable_parentheses 1`. A build without `SQLITE_ENABLE_FTS3_PARENTHESIS` will not parse `"a" OR "" bbb NEAR 0 NEAR 1` the same way. The Dockerfile passes the flag because the CLI does. Leave it off and you lab a syntax error.

Treating ASan leak detection as the oracle. `detect_leaks=0`. This is a use-after-free, not a leak. Leak reports on sqlite3 shutdown are noise. `abort_on_error=1` is the halt that turns the READ into rc 134.

Putting a network listener on sqlite3. The library does not listen. The compose port is health. `exec sqlite3 :memory:` is the oracle. Publishing 18880 without `health.py` just makes `run.sh` wait for a process that will never answer HTTP.

A reverse shell. Theatre. The oracle is an ASan stack that names `fts3SnippetAdvance` and `fts3EvalRestart`.

10 What an attacker can do

Reach a process that will run attacker SQL against an FTS3/4 table, or open an untrusted `.db` whose schema already has one. MATCH plus `snippet()`. Incremental phrase under OR/NEAR.

`fts3EvalRestart` frees a phrase doclist. `snippet()` still walks the cached poslist pointers. Heap use-after-free. Crash. On an allocator without ASan, the same READ

is a dangling pointer into recycled heap. This pack does not demonstrate that turning into code execution. Class is UAF, not RCE.

It is local. It is not unauthenticated HTTP. SQLite's own security model still treats that SQL as a security input, which is why FTS3 has a CVE history at all. Embedders that take MATCH from a search box and call `snippet()` for the UI are the boring case. Shadow tables are the Magellan case. Both are in scope.

I am not going to print a one-liner you can paste at someone else's `sqlite3`. The missing second iterate in `fts3BestSnippet` is the useful part. If you own the binary, run the proof of concept against loopback.

11 The fix

Iterate phrases twice in `fts3BestSnippet`. First pass lets `sqlite3Fts3EvalPhrasePoslist` restart whatever needs restarting. Second pass fills `pHead` / `pTail` from doclists that still exist. That is trunk e4635c4153. Cherry-pick is already on `branch-3.53` as 9a367c177f. 3.53.4 does not have it.

Until you have that check-in, do not hand untrusted MATCH plus `snippet()` to FTS3/4 on this GA. Compile without FTS3/4 if you do not need it. The CLI default does need it, which is why the leftover is typical extra and not "operator opted into a research flag."

Do not take "Dan fixed it in August" as a reason to skip the 3.53.4 binary. August is after July. Leftovers look like that when they ship.

12 References

- Proof of concept: abraxas/sqlite-fts3-snippet · sqlite-fts3-snippet-Abraxas-Labs.py

- Lab: `lab/` · Dockerfile · docker-compose.yml · run.sh · poc.py · health.py
- @abraxas_null · github.com/abraxas · abraxaslabs.tech · abraxas.null@proton.me
- CWE-416 · 8.1 High
- Tree: sqlite/sqlite version-3.53.4 (`b09c88c14082339b66c7b7158d609a771e64ca69`) · canonical sqlite.org/src · SOURCE_ID `2026-07-24 19:02:57 bf7c7f30031888f4e796e429ab3978879485813aaca6f641c7b33e4e09459bcc`
- `fts3_snippet.c` `fts3BestSnippet` / `fts3SnippetFindPositions` / `fts3SnippetAdvance` · `fts3.c` `fts3EvalRestart` / `fts3EvalInvalidatePoslist` / `sqlite3Fts3EvalPhrasePoslist`
- Forum 2026-08-17T09:18:24Z (Marc Heuse) · trunk fix e4635c4153 · `branch-3.53` 9a367c177f
- 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**

Home Research Open Source Reading Room About GitHub @abraxas_null Console

---

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

© 2026 Abraxas Labs