0-DAY NO CVE HIGH PUBLIC

SQLite Session, same-name changeset nCol heap write

SQLite 3.53.4 Session still keys apply state on table name. A later changeset block with the same name and a larger nCol skips sessionTableInfo reset and memset()s past aUpdateMask. Labbed ASan WRITE of size 12 in sessionUpdateFind. Compile extra. CVE-2026-50812 NPD is closed. No CVE yet.

NAME SQLite Session, same-name changeset nCol heap write	TYPE 0-day analysis
CVE n/a	CVE RISK high
DISCLOSURE STATUS public	VENDOR SQLite (Hwaci)
AFFECTED SQLite 3.53.4 (b09c88c) compiled -DSQLITE_ENABLE_SESSION. Default amalgamation without SESSION is not this leftover. Untrusted sqlite3changeset_apply*.	PUBLISHED 09 Oct 2026
UPDATED 09 Oct 2026	TAGS 0-day, sqlite, session, heap-oob-write, cwe-787

TL;DR

Lab files and proof-of-concept **here**.

01 The table name was the cache key. The column count was not.

I am @abraxas_null. The proof of concept is on GitHub: **abraxas/sqlite-session-ncol** (loopback client). The lab stack is `lab/` : Dockerfile, docker-compose.yml, run.sh, poc.py, poc.c, health.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` . Compile extra `-DSQLITE_ENABLE_SESSION` . No CVE yet. CWE-787. High 8.1 (`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`). Local. Heap OOB write. Untrusted changeset blob through `sqlite3changeset_apply*` . Default amalgamation without SESSION is not this leftover.

SQLite Session is the sync API. You record writes, you ship a changeset, the other side applies it. The format is a sequence of table blocks. Each block has a name, a column count, a PK mask, then INSERT / UPDATE / DELETE records. Apply is supposed to look at the live schema, refuse a block that asks for more columns than the table has, and cache prepared statements for that table until the name changes.

The cache key is the name. Only the name. A second block that says `t` again, with a larger `nCol` , does not re-enter `sessionTableInfo` . The UPDATE helper then `memset` s a mask sized from the new count over an allocation sized from the old one. Two blocks. Same name. First `nCol=2` , then `nCol=64` . ASan WRITE of size 12 in `sessionUpdateFind` .

That is the leftover. Not CVE-2026-50812. That NPD is closed on this pin.

02 Why SQLite

I had just finished unpublished Crit/High hunts of Caddy, lighttpd, and OpenLiteSpeed. HTTP servers have a shape: parsers, CGI, TLS, a leftover of last year's 400 list. SQLite is the other shape. A C library that treats untrusted SQL and untrusted database files as first-class inputs, documents that on sqlite.org/security.html, and still ships compile extras that talk to the heap in ways the default amalgamation never will.

I only wanted Critical or High. Local is in-scope. Medium stays on the floor unless it chains. A finding on this hunt needed memory corruption, crash, or heap disclosure on untrusted SQL or an untrusted `.db`, or a leftover of a named 3.53.x / 3.54 fix that still fires on latest GA. `load_extension` is off. I am not filing "ATTACH any path." I am not filing a log callback.

Session is not the default library. It is the extra you turn on when you want changeset apply. Apps that sync, replicate, or ingest a peer's blob compile `-DSQLITE_ENABLE_SESSION` and `-DSQLITE_ENABLE_PREUPDATE_HOOK`. That is a smaller install base than FTS5. It is also a trust-boundary extra: the blob is supposed to be hostile.

03 Pinning 3.53.4

Latest GA. Not an RC. Not the draft.

sqlite.org listed 3.53.4 (2026-07-24). 3.54.0 was still a draft with a planned 2026-10-15. Pinning a draft is how you "discover" bugs the trunk already closed. The pin is the tag people actually ship.

Canonical tree is Fossil at sqlite.org/src. GitHub sqlite/sqlite is a mirror. The Git tag `version-3.53.4` lands on `b09c88c14082339b66c7b7158d609a771e64ca69`. `SQLITE_SOURCE_ID` is the Fossil fingerprint, not the Git SHA. If your binary prints a different SOURCE_ID you are not on this pin.

I built the lab from the official `sqlite-src-3530400.zip`, not from a random GitHub tarball, so the amalgamation and the Session extra come from the same release the

download page names.

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. On 3.53.4 the interesting ones were already closed:

- CVE-2026-11822 / CVE-2026-11824 FTS5 heap overflow when defensive mode is off. Duplicate CVEs. Fixed 3.53.2. `fts5LeafRead` rejects `szLeaf<4`.
- CVE-2026-50812 Session NPD in `sqlite3changeset_apply_v3` on a malformed changeset. Medium CWE-476. Fixed 3.53.2 (`e807d4e379` / Git `b869ed6`). `sessionBindRow` treats a missing value as `SQLITE_CORRUPT`. Debian marks 3.53.4-2 fixed. Optional `-DSQLITE_ENABLE_SESSION`. Closed on this pin.
- CVE-2025-3277 / CVE-2025-29087 `concat_ws` integer overflow. Fixed 3.49.1.
- CVE-2025-29088 LOOKASIDE OOB. `setupLookaside` i64 clamp is in this tree.
- CVE-2025-70873 zipfile CLI OOB. CLI extra. Fixed 3.52.0.
- CVE-2024-0232 JSON UAF. Fixed 3.43.2. The subtype/UPLUS walk on this pin is the Fossil parent of the tag.
- CVE-2023-7104 Session on a corrupted changeset. Fixed 3.43.1. Iterator parse uses `sessionVarintGetSafe`. `nCol>65536` is corrupt.
- Ada Logics `sqlite3_rsync` stack smash. `readPow2` `x>16` is in this pin. Not this leftover.

- FTS3 NEAR negative-delta overwrite and signed overflow. In this pin.
- `load_extension` SQL and C API deny unless the app sets the enable bits. Function is `DIRECTONLY|UNSAFE`.

CVE-2026-51296 and friends are the JFrog "critical SQLite CVEs" that SQLite and JFrog later treated as unreproducible LLM slop. I left them.

Vendor table: sqlite.org/cves.html. Developers fix in hours and do not always mint a CVE. Public forum leftovers of later trunk are a different list from the CVE table. That is where this one lived.

05 Twenty surfaces, one Session leftover

I split the tree. Parser and VDBE. Btree 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` / VFS. C API. Recover and misc extras. Schema side effects.

Most of that is SQLite doing what 3.53.4 said it would do, or a leftover that is not this post.

Default `load_extension` stays off. TRUSTED_SCHEMA stays ON. Typical amalgamation extras are FTS5, JSON, rtree. Session is not typical. Official WASM is another extra. The CLI is another extra. I am not bundling those into one report.

RBU July-2026 heap over-reads are closed on this pin. `rbuDeltaApply` terminator, mask length, `nRow<0` are gated. Session rebase integer overflow from the public 2026-06-26 report is closed (`sessionBufferGrow` takes `i64`). Extended-result-code conflict mishandling is leftover correctness: UNIQUE becomes `SQLITE_CONSTRAINT_UNIQUE`, retry skips, savepoint leaks. Not a shown UAF. Below this bar.

The Session hunt still had a High. Name as cache key. Column count as an afterthought.

06 CVE-2026-50812 is the NPD. This is the write.

50812 is a missing value. Apply walks an UPDATE, `sqlite3changeset_new` / `_old` yields NULL, `sessionBindValue` used to call `sqlite3_value_type(NULL)`. 3.53.2 made that `SQLITE_CORRUPT`. The bind path on this pin checks the pointer. I grepped it. I did not re-file it.

The leftover is the table-block cache in `sessionChangesetApply`. Forum 2026-08-16T10:15:25Z (vanhauser). Reporter ASan on 3.54.0 (SOURCE_ID 2026-07-31). This pin is 3.53.4 (2026-07-24). Dan: trunk `3f57024aec` / branch-3.53 `a1dd7123d8` (2026-08-17). After this GA. No CVE on the forum thread.

That is the leftover class I wanted. A named public bug, a trunk fix, latest GA still open, High if it is still a heap write.

07 Apply keys ctx on the name

`sessionChangesetApply` walks `sqlite3changeset_next`. For each record it asks the iterator for the table name, column count, and op. Then this:

C Copy

```c
if( zTab==0 || sqlite3_strnicmp(zNew, zTab, nTab+1) ){
    /* free cached UPDATE stmts, free aUpdateMask, sessionTableInfo,
       refuse sApply.nCol < nCol, refuse PK mismatch, rebuild INSERT/DELETE */
}
```

`sqlite3session.c:5619`. The reset, the schema check, the `nCol<nCol` refuse, the PK memcmp, all live inside that `if`. Same name: skip the block. `schemaMismatch` stays whatever it was. `sApply.aUpdateMask` stays the first-block pointer.

`sessionUpdateFree` is the function that actually `sqlite3_free`s `aUpdateMask` and nulls it. It is only called from that name-changed arm. Same name never frees the mask.

Then an UPDATE hits `sessionUpdateFind`:

```c
int nCol = pIter->nCol;
int nU32 = (pIter->nCol+33)/32;

if( p->aUpdateMask==0 ){
  p->aUpdateMask = sqlite3_malloc(nU32*sizeof(u32));
}

if( rc==SQLITE_OK ){
  memset(p->aUpdateMask, 0, nU32*sizeof(u32));
  for(ii=0; ii<pIter->nCol; ii++){
    if( sessionChangesetNew(pIter, ii) ){
      p->aUpdateMask[ii/32] |= (1<<(ii%32));
```

`sqlite3session.c:4381-4412`. Allocate only when the pointer is NULL. Size the `memset` and the bit-set from `pIter->nCol`, which is the *current* block. First block `nCol=2`: `nU32 = (2+33)/32 = 1`, malloc 4 bytes. Second block `nCol=64`: `nU32 = (64+33)/32 = 3`, `memset` 12 bytes over a 4-byte heap object.

WRITE of size 12. That is not a coincidence. That is `3 * sizeof(u32)` from a 64-column mask written onto a 2-column allocation. The bit-set loop and the patchset high bit can write further. Same-name later INSERT/DELETE also index `p->abPK[i]` / `p->azCol[i]` with iterator `nCol`.

All six apply entry points share this: `sqlite3changeset_apply`, `_v2`, `_v3`, and the `_strm` variants. They funnel through `sessionChangesetApplyV23` into `sessionChangesetApply`.

A well-formed single-block UPDATE against a 2-column table is fine. The schema check runs. `nCol=2` matches. `apply-rc=0`. That is the control.

A well-formed single-block UPDATE that *declares* 64 columns against a 2-column table is also fine, in the other direction. `sApply.nCol<nCol` sets `schemaMismatch`. No write. That is the wrong experiment. The leftover is two blocks, same name, growing `nCol`.

08 The changeset is not SQL

Session does not parse `UPDATE t SET ...`. The blob is a tiny binary format. Table header `'T'`, `nCol`, PK bytes, name, NUL, then `SQLITE_UPDATE` and old/new values. The lab C file writes that by hand. First block: table `t`, 2 columns, PK on column 0, UPDATE id 1, text `old` → `new`. Second block: table `t` again, 64 columns, PK still column 0, UPDATE id 1 → 2, the extra fields NULL.

Python `urllib` is not in this story. There is no HTTP API. The "listener" on `127.0.0.1:18900` is a health check so `run.sh` knows the ASan binary finished compiling. The trigger is `docker compose exec lab sqlite-session-ncol trigger`.

I am not going to print a 242-byte changeset you can paste into someone else's `sqlite3changeset_apply`. The missing name-vs-nCol identity is the useful part. If you own the process, run the proof of concept against loopback.

09 The lab

Host is macOS. One container.

Image from `debian:bookworm-slim`. Dockerfile curls `https://www.sqlite.org/2026/sqlite-src-3530400.zip`, `./configure`, `make sqlite3.c`, then:

PLAIN TEXT Copy

```
cc -O1 -g -fsanitize=address -fno-omit-frame-pointer
    -DSQLITE_ENABLE_SESSION -DSQLITE_ENABLE_PREUPDATE_HOOK
```

```
    poc.c sqlite3.c
```

Compose project `sqlite-session-ncol`. Published `127.0.0.1:18900`. `ASAN_OPTIONS=detect_leaks=0:abort_on_error=1:halt_on_error=1`. Health body `ok SQLITE-SESSION-NCOL-WITNESS`. The binary prints `version=` and `source-id=` so the PoC can refuse a wrong pin before it claims SUCCESS.

The PoC:

01 GET health. 200 and witness, or FAIL.

02 `sqlite-session-ncol version`. Must be `3.53.4` and SOURCE_ID starting `2026-07-24`.

03 `sqlite-session-ncol control`. Single 2-col UPDATE. Must exit 0, `apply-rc=0`, no ASan.

04 `sqlite-session-ncol trigger`. Two blocks, nCol=2 then 64. Must be ASan `heap-buffer-overflow`, `WRITE of size N`, sink `sessionUpdateFind` / `aUpdateMask` / `sqlite3session.c`.

Last run:

PLAIN TEXT Copy

```
control-exit=0 apply-rc=0 asan=0
n-changeset=242 first-nCol=2 second-nCol=64
ERROR: AddressSanitizer: heap-buffer-overflow
WRITE of size 12
#1 ... in sessionUpdateFind ... sqlite3.c:238155
SUMMARY: AddressSanitizer: heap-buffer-overflow ... __interceptor_memset
trigger-exit=134 apply-rc=none asan=1 write-size=12
SUCCESS sqlite-session-ncol how=asan-heap-write-sessionUpdateFind write-size=12 c
```

Amalgamation line 238155 is the Session extra folded into `sqlite3.c`. The source file on the pin is still `ext/session/sqlite3session.c`. `sessionUpdateFind` is the function. `memset` is the write. Size 12 is the 64-column mask.

Replay:

PLAIN TEXT Copy

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

Bind it to loopback. The loopback client is `lab/run.sh` with a banner on it.

10 Wrong turns, in the order I made them

Pinning 3.54.0. It looks like "latest." It is a draft. The vanhauser report is against 3.54.0 (2026-07-31). Trunk closed it on 2026-08-17. Latest GA is 3.53.4. The leftover is the GA, not the draft.

Treating GitHub HEAD as the pin. `sqlite/sqlite` is a Fossil mirror. SOURCE_ID is the fingerprint. `b09c88c` is the Git tag. If those disagree you are in a rebase, not a review.

Re-filing CVE-2026-50812. Missing old/new value, NPD, 3.53.2, Medium. The bind NULL path is closed. The name-keyed mask is not that bug. Same extra, different sink, different year.

Treating default `libsqlite3` as in-scope for this extra. Stock amalgamation does not define `SQLITE_ENABLE_SESSION`. The lab `#error`s if you forget the flag. A FAIL on a default build is a lab of the compile option, not of `sessionUpdateFind`.

One table block with nCol=64. Schema check fires. `sApply.nCol<nCol`. `schemaMismatch=1`. No `sessionUpdateFind` write. That proves the first-block path works. The leftover is the second block that never takes that path.

Two blocks with different names. Name change calls `sessionUpdateFree`, frees `aUpdateMask`, re-runs `sessionTableInfo`. That is the *working* arm. The bug is the arm you do not take.

A huge nCol and a 2GiB blob. There is a separate tblhdr `int` overflow on `nCol=65536` plus a table name that fills most of an `int`-sized changeset. Different class, not labbed, not this write-of-size-12. Do not mix them.

Python-only changeset. struct packing in Python is how you get off-by-one in the PK mask and then spend a day debugging your generator. The oracle binary is C against the amalgamation. Python drives compose exec and greps ASan.

Compiling without ASan and calling exit 134 a crash finding. 134 is ASan abort. Without ASan the same `memset` is a silent heap smash, which is worse and also not an oracle you can print. The SUCCESS line records `how=asan-heap-write-sessionUpdateFind write-size=12` so nobody can pretend this was a NULL deref.

Claiming RCE. A 12-byte `memset` past a 4-byte heap object is a heap OOB write. It is not a demonstrated write-what-where. It is not instruction-pointer control. Heap grooming does not belong in this pack. ASan WRITE-of-size-12 is the oracle. Stop there.

Calling RBU this bug. RBU is another extra. Hostile rbu db leftovers on this pin are closed or below bar. Session apply of a changeset is the API.

Filing "untrusted SQL crashes SQLite." The security model already says untrusted SQL is in-scope and then spends a lot of code making that not a heap write. Session apply is not SQL. It is a second parser with a second cache key.

A reverse shell. Theatre. The oracle is an ASan line on loopback.

11 What an attacker can do

Reach a process that compiled SQLite with `-DSQLITE_ENABLE_SESSION` and applies a changeset from a peer it does not already trust. `sqlite3changeset_apply`, `_v2`,

`_v3`, or the streaming variants. No login inside SQLite. The privilege is "the host called apply on my blob."

Send two table blocks. Same name. First `nCol` is a count the live schema accepts. Second `nCol` is larger. Then an UPDATE. Apply skips `sessionTableInfo`. `sessionUpdateFind` `memset`s past `aUpdateMask`. Heap OOB write as the apply UID.

Not a default amalgamation unless SESSION is on. Not CVE-2026-50812. Not RCE in this lab. Crash or heap corruption in the process that called apply. Sync daemons, replica appliers, and anything that takes a changeset off the wire or out of a file are the victim model. A local `sqlite3` CLI that never calls the Session API is not.

I am not going to print a changeset you can paste at someone else's replica. The missing reset on same-name `nCol` growth is the useful part. If you own the process, run the proof of concept against loopback.

12 The fix

Stop treating the table name as the identity of the validated apply state. Each table block should have its declared column count and PK layout checked before any operation. If a same-name block changes either value, reject it, or discard `SessionApplyCtx` the way a new name already does: `sessionUpdateFree`, `sessionTableInfo`, refuse `nCol` larger than the schema.

Trunk `3f57024aec` / branch-3.53 `a1dd7123d8` (2026-08-17) is that class of fix. It is after this GA. 3.54.0 was still draft when I pinned 3.53.4.

Until then, do not apply an untrusted changeset on 3.53.4 Session. Build without `-DSQLITE_ENABLE_SESSION` if you do not need the extra. If you need the extra, take the trunk change or refuse a blob that repeats a table name with a different `nCol`.

Do not take "50812 is fixed" as a reason to skip Session apply. The NPD is one malformed record. The write is two well-typed blocks and a cache key that is too small.

13 References

- Proof of concept: abraxas/sqlite-session-ncol · sqlite-session-ncol-Abraxas-Labs.py
- Lab: `lab/` · Dockerfile · docker-compose.yml · run.sh · poc.py · poc.c · health.py
- @abraxas_null · github.com/abraxas · abraxaslabs.tech · abraxas.null@proton.me
- CWE-787 · High 8.1
- Tree: sqlite.org/src tag version-3.53.4 (`b09c88c14082339b66c7b7158d609a771e64ca69`) · SOURCE_ID `2026-07-24 19:02:57 bf7c7f30031888f4e796e429ab3978879485813aaca6f641c7b33e4e09459bcc` · `sessionChangesetApply` name check · `sessionUpdateFind` mask
- Forum 2026-08-16T10:15:25Z · trunk fix `3f57024aec` · CVE-2026-50812 session NPD closed 3.53.2 · sqlite.org/cves.html · sqlite.org/security.html

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