E08. Target-acquisition code and database aggregates

Ilja Krasnjanski
24 October 2026 · Edited evidence report
Historical evidence through 24 July 2026; later engineering checks identified separately.

This report synthesizes the cited case records. Exhibits distinguish retained artifact excerpts from secondary analytical records and claims whose original inputs are unavailable. Excerpts may be redacted or reformatted; extraction methods and limitations are stated locally. Fingerprints identify inputs, not the truth of their claims. Bracketed S labels identify the analysis records listed at the end.

Collection implementation

The recovered harvester searches repositories by star range, visits branches and extracts both author and committer email fields, lowercasing and trimming them before filtering. The source record describes up to twenty branches per repository, worker-specific ranges, progress tracking and rate-limit handling. Scheduler and merge components consolidate worker results into two SQLite pools consumed by sender configurations. A saved log describes an April 2026 run; its throughput is a log claim, not an independently measured acquisition rate. [S1]

Aggregate method

The recorded verification opens the held databases read-only. The safe pool is the main database; the antispam pool is attached as `other`. Both expose an `emails` relation with `email` and `repo` fields. These aggregate queries compare already-stored values without additional lowercasing, canonicalization or person-level identity resolution; ingestion itself lowercases and trims addresses. No row-level contact data is included. The following results were reproduced in the read-only, aggregate-only check recorded on 24 October 2026 in Exhibit E08-5. [S2, S3]

`SELECT count(*) FROM main.emails;` — 79,698.

`SELECT count(*) FROM other.emails;` — 62,667.

`SELECT count(*) FROM main.emails s JOIN other.emails a ON s.email = a.email;` — 0.

`SELECT count(*) FROM (SELECT email FROM main.emails UNION SELECT email FROM other.emails);` — 142,365.

`SELECT count(*) FROM (SELECT repo FROM main.emails UNION SELECT repo FROM other.emails);` — 27,698.

Interpretation

The pools are disjoint under the tested address comparison, while their repository sets overlap. An earlier total of 20,880 repositories represented only the safe pool and is not the combined result. The combined count measures distinct stored repository values, not verified active repositories. [S1, S2]

The analysis note describes sender configurations referencing both pools with differing intervals, cooldowns and warm-up settings. Those configurations are not retained among the exhibit inputs, so this linkage remains a source-note finding. It does not prove that all stored contacts received a message. The labels safe and antispam are the tool's categories, not independent assessments of recipient risk.

Limits and handling

The unit of analysis is an address string. Aliases, case variants and multiple addresses per person remain unresolved; 142,365 is not a victim or infection count. The two databases remain withheld because they contain harvested contact information. Their source hashes identify the particular inputs but do not make the totals independently reproducible without authorized access. [S4, S5]

Exhibits

Exhibit E08-1. Harvester: up to twenty branches per repository, commit-author email extraction

Artifact: Github-Fetch-Tool worker — `code/1_main.py` · 15996 bytes · SHA-256 `a5fca8cb06a62e8bcbe88c3966de608afc5c861359e02517135be81815cb28bf`

Location: lines 11, 228-261 (`get_branches`), 272-312 (`get_commit_emails`)

Extraction: `sed -n '11p;228,312p' code/1_main.py`

Supports: E08 "searches repositories by star range, visits branches and extracts commit-author email fields ... up to twenty branches per repository".

```
MAX_BRANCHES_PER_REPO = 20
...
def get_branches(owner, repo, headers):
    """Return a list of branch names for a repo (capped at MAX_BRANCHES_PER_REPO)."""
    ...
        if len(branches) >= MAX_BRANCHES_PER_REPO:
            branches = branches[:MAX_BRANCHES_PER_REPO]
...
            for c in commits:
                commit_data = c.get("commit", {})
                author_email = (commit_data.get("author") or {}).get("email", "")
                committer_email = (commit_data.get("committer") or {}).get("email", "")
                for raw in (author_email, committer_email):
                    cleaned = raw.lower().strip()
                    if cleaned and not is_github_email(cleaned):
                        emails.add(cleaned)
```

Confirms the 20-branch cap and that commit author/committer email fields are collected. No tokens present (the sample's config token field is the empty string). No redaction needed.

Exhibit E08-2. Worker-specific star ranges and the emails(email, repo) schema

Artifact: Github-Fetch-Tool worker configs — `code/config.json`, `code/2_config.json`, `code/3_config.json` (103 bytes each) · SHA-256 `8a6e5865ff398087a272d2316b410ca185d94d7510eeecd56e4e19102d73c053`, `35472cfd1a99c046c57769e2da297255bbe2a96a445e28a4c5aad35c0ef9426b`, `dc4825bea2866f391e25ea3555d029f1019668bcb9d76c9c62b8b890e146e5bf`

Location: full files; plus `CREATE TABLE` at `code/1_main.py` lines 326-331

Extraction: `cat code/config.json code/2_config.json code/3_config.json`

Supports: E08 "worker-specific ranges"; the `emails` relation with `email` and `repo` fields.

```
{ "github_token": "", "queries": ["stars:101..137"], "per_page": 10, "max_page": 100 }
{ "github_token": "", "queries": ["stars:138..174"], "per_page": 10, "max_page": 100 }
{ "github_token": "", "queries": ["stars:175..210"], "per_page": 10, "max_page": 100 }
```

```
CREATE TABLE IF NOT EXISTS emails (email TEXT PRIMARY KEY, repo TEXT)
```

Each worker carries its own star band. Token fields are empty in the retained sample (cleared before commit). Three config files pretty-printed to one line each for width; contents unchanged.

Exhibit E08-3. Merge into two pools by deliverability (safe vs antispam)

Artifact: Github-Fetch-Tool — `code/merge.py` · 3009 bytes · SHA-256 `cb642220331f16af6b31d2b9bc9114cb61df6487512f1654ba6081de310a1960`

Location: lines 5-23, 59-73

Extraction: `sed -n '5,23p;59,73p' code/merge.py`

Supports: E08 "Scheduler and merge components consolidate worker results into two SQLite pools"; the safe/antispam labelling.

```
ANTISPAM_DB = "results_antispam.db"
SAFE_DB = "results_safe.db"
# Providers with strict anti-spam filters that block bulk sending
ANTISPAM_DOMAINS = {
    "gmail.com", "google.com", "googlemail.com",
    "outlook.com", "hotmail.com", "live.com", "msn.com",
    ...
}
def is_antispam(email: str) -> bool:
    domain = email.rsplit("@", 1)[-1].lower()
    return domain in ANTISPAM_DOMAINS
```

Freemail domains route to the antispam pool, everything else to the safe pool; INSERT OR IGNORE dedupes on the `email` primary key. This is the tool's own category, not a recipient-risk judgment.

Exhibit E08-4. Scheduler log of the April 2026 run (throughput is a log claim)

Artifact: Github-Fetch-Tool — `scheduler.log` · 3334 bytes · SHA-256 `e1cc50c11671ebf43a0310bf9085e9c3388341cdb62fde443c86674f92d82091`

Location: lines 39-42 (the 2026-04-19→04-20 merge)

Extraction: `sed -n '39,42p' scheduler.log`

Supports: E08 "A saved log describes an April 2026 run; its throughput is a log claim".

```
[2026-04-20 00:00:18 UTC]   merge: New anti-spam emails added: 57567
[2026-04-20 00:00:18 UTC]   merge: New safe emails added: 72201
[2026-04-20 00:00:18 UTC]   merge: Total unique emails in results_antispam.db: 62667
[2026-04-20 00:00:18 UTC]   merge: Total unique emails in results_safe.db: 79698
```

Self-reported run totals; not an independently measured acquisition rate. The byte `0xef` (" ") in some log lines is a mojibake arrow in the original and is not reproduced here. No redaction needed.

Exhibit E08-5. Aggregate queries over the two withheld pools (no rows shown)

Artifact: safe pool `results_safe.db` · 6922240 bytes · SHA-256 `a84cee4ce20e0d40177b750a2692acd37ae6cd30a4e303790c6eb1ecf6348187`; antispam pool `results_antispam.db` · 5496832 bytes · SHA-256 `7eff7dac7119b6b4462e685855a3e81269ecf9ecb1f392fa4e51282825d4429c`

Location: aggregate SQL only; `other` = antispam attached read-only

Extraction: `sqlite3 -readonly results_safe.db` with `ATTACH DATABASE 'file:results_antispam.db?mode=ro' AS other;`

Supports: E08 counts (79,698 / 62,667 / 0 overlap / 142,365 union / 27,698 repo union).

```
SELECT count(*) FROM main.emails;                                          -- 79698
SELECT count(*) FROM other.emails;                                         -- 62667
SELECT count(*) FROM main.emails s JOIN other.emails a ON s.email = a.email;   -- 0
SELECT count(*) FROM (SELECT email FROM main.emails
                      UNION SELECT email FROM other.emails);                -- 142365
SELECT count(*) FROM (SELECT repo FROM main.emails
                      UNION SELECT repo FROM other.emails);                 -- 27698
```

Re-ran 2026-10-24 with `sqlite3 -readonly`, aggregate-only. Reproduces every count in E08 exactly. No row, email or repo value was selected or printed.

Exhibit E08-6. Per-pool distinct repository counts corroborating the 20,880 correction

Artifact: same two pools as E08-5

Location: aggregate SQL only

Extraction: `sqlite3 -readonly results_safe.db` (antispam attached as `other`)

Supports: E08 "An earlier total of 20,880 repositories represented only the safe pool".

```
SELECT count(DISTINCT repo) FROM main.emails;   -- 20880   (safe pool)
SELECT count(DISTINCT repo) FROM other.emails;  -- 21366   (antispam pool)
-- schema check (no rows): CREATE TABLE emails (email TEXT PRIMARY KEY, repo TEXT) in both
```

Safe pool alone has 20,880 distinct repos; the union (E08-5) is 27,698, so the two repo sets overlap. Confirms the correction and that `email`/`repo` are the only columns. Aggregate-only.

Claims without a retained primary artifact

- Sender configuration ("references both pools with differing intervals, cooldowns and warm-up settings"): the `Email-Sender-*/sender_config.json` files are described in the analysis note and the public-repo report (`19-github.txt`) but are not retained as files in the workspace.

Exhibit inputs

Retained inputs identified by size and SHA-256. These include artifacts and analytical records; each exhibit identifies its evidence type and extraction limitations. The files themselves are not distributed.

Github-Fetch-Tool worker — code/1_main.py · 15,996 bytes

`SHA-256 a5fca8cb06a62e8bcbe88c3966de608afc5c861359e02517135be81815cb28bf`

Github-Fetch-Tool worker config — code/config.json · 103 bytes

`SHA-256 8a6e5865ff398087a272d2316b410ca185d94d7510eeecd56e4e19102d73c053`

Github-Fetch-Tool worker config — code/2_config.json · 103 bytes

`SHA-256 35472cfd1a99c046c57769e2da297255bbe2a96a445e28a4c5aad35c0ef9426b`

Github-Fetch-Tool worker config — code/3_config.json · 103 bytes

`SHA-256 dc4825bea2866f391e25ea3555d029f1019668bcb9d76c9c62b8b890e146e5bf`

Github-Fetch-Tool — code/merge.py · 3,009 bytes

SHA-256 cb642220331f16af6b31d2b9bc9114cb61df6487512f1654ba6081de310a1960

Github-Fetch-Tool — scheduler.log · 3,334 bytes

SHA-256 e1cc50c11671ebf43a0310bf9085e9c3388341cdb62fde443c86674f92d82091

Harvested safe pool (withheld PII) — results_safe.db · 6,922,240 bytes

SHA-256 a84cee4ce20e0d40177b750a2692acd37ae6cd30a4e303790c6eb1ecf6348187

Harvested antispam pool (withheld PII) — results_antispam.db · 5,496,832 bytes

SHA-256 7eff7dac7119b6b4462e685855a3e81269ecf9ecb1f392fa4e51282825d4429c

Source records and fingerprints

SHA-256 identifies the reviewed source bytes, not acquisition time or every artifact described. Restricted originals are not reproduced.

[S1] Target-acquisition analysis, 24 July 2026 · 6,162 bytes

SHA-256 cf1dca04cc4435a57cef82991a6cd6a2576b6043b260e384584f84efe19e11a4

[S2] Edited preprint manuscript · 54,752 bytes

SHA-256 dc5d8abe46fcd33e9a28ccc859778e22ced661beff2d559a072bfb04f9d131dc

[S3] Preprint validation record · 10,642 bytes

SHA-256 d73c43a5f61366ddf82a755fd473a0f72437731f9dee01ee325b0314e8c9d3e9

[S4] Restricted safe-pool SQLite input · 6,922,240 bytes

SHA-256 a84cee4ce20e0d40177b750a2692acd37ae6cd30a4e303790c6eb1ecf6348187

[S5] Restricted antispam-pool SQLite input · 5,496,832 bytes

SHA-256 7eff7dac7119b6b4462e685855a3e81269ecf9ecb1f392fa4e51282825d4429c