EmailFinder — Comprehensive Technical Overview

Explainable Business Email Discovery Without SMTP

A client-readable and engineer-ready reference for the discovery waterfall, DNS and public-web signals, candidate scoring, CLI workflows, test evidence, privacy boundaries, responsible use, operational risks, and developer handoff.

Python 3.10+ | HTTP + DNS Signals | Four CLI Commands | Explainable Scoring

Updated October 2026 | Original source review July 2026

Developed by Zahidul Islam

Latest verification — 14 October 2026

The detailed technical sections below remain current. The local repository has no source commit after its 16 April 2026 initial release.

Fresh local checks: all 23 collected pytest cases passed with the declared dnspython dependency available. CLI help displayed the find, verify, scrape, and batch commands.

Setup and evidence limit: a separate Python environment without dnspython could not import the package. No live DNS, website, Gravatar, or GitHub lookup was repeated for this update. The July 2026 review remains below as dated historical evidence.

Current Evidence and Scope Note — July 2026

This edition was reconciled against the current Python implementation and deterministic tests. It describes a local research assistant, not a mailbox-existence authority, deliverability guarantee, consent decision, or hardened multi-user service.

Verified repository snapshot

- Package version 1.0.0 with one declared external dependency.
- Four CLI commands: find, verify, scrape, and batch.
- Twenty Python files with 60 assertion statements.
- All four deterministic test scripts passed on 2026-07-21.
- Live DNS, website, Gravatar, and GitHub integrations were not exercised in this documentation pass.

1. Reading Guide and Evidence Policy

This dossier describes EmailFinder for both client and developer review. Client sections explain what the product can and cannot conclude, what external signals it uses, how results should be interpreted, and what responsible-use controls apply. Developer sections document the command surface, module ownership, discovery flow, scoring, failures, tests, privacy boundaries, and operational handoff.

“Implemented” means behavior was found in the inspected Python source. “Locally verified” means a deterministic test was executed during the July 2026 review. External DNS, website, avatar, and code-hosting lookups were not exercised during this documentation pass. A discovered candidate is a research lead, not proof of mailbox ownership, deliverability, permission to contact, or consent.

2. Executive Overview

EmailFinder is a local Python command-line research assistant for discovering and evaluating publicly inferable business email candidates. It combines domain mail configuration, public website text, conventional address patterns, and limited external signals into explainable ranked results. It does not send mail and deliberately avoids SMTP mailbox probing.

The implementation is compact and auditable: four commands route through shared DNS, scraping, verification, pattern, scoring, and formatting modules. Its confidence values are deterministic heuristic scores. They communicate why one candidate ranked above another under the implemented rules; they are not calibrated probabilities and should never be presented as guaranteed accuracy.

3. Product Purpose and Boundaries

The product supports legitimate research where a reviewer has a person name, organization domain, or public business website and needs to identify plausible public contact routes. It is useful for organizing visible addresses, generating conventional business patterns, and distinguishing stronger public evidence from weaker inference.

It does not establish that an inbox exists, that a person owns it, that a message will be delivered, or that outreach is lawful or appropriate. It does not verify catch-all behavior, negotiate with mail servers, bypass access controls, authenticate users, manage campaigns, or send messages. Consent, purpose, suppression, retention, and jurisdiction-specific compliance remain outside the scoring engine and must be enforced by the operator's process.

4. Verified Implementation Snapshot

- Package: version 1.0.0 for Python 3.10 or later.
- Dependency surface: one declared external package, used for DNS operations.
- Source surface: 20 Python files in the inspected project.
- Command surface: find, verify, scrape, and batch.
- Test surface: four deterministic test programs containing 60 assertion statements.
- Local evidence: all four deterministic test programs passed on 2026-07-21.
- External evidence boundary: no live DNS, website, Gravatar, or GitHub integration call was performed for this documentation update.
- Delivery boundary: the code contains no SMTP sending workflow.

5. Architecture Overview

EmailFinder is organized as a CLI shell around small research services. Argument parsing selects a command and validates required inputs. Command handlers call shared domain normalization, DNS, scraping, candidate generation, external-signal, scoring, and output modules. Network behavior is concentrated in helpers with a fixed user agent, request timeout, and process-level pacing.

The program is stateless across invocations unless the operator redirects output or uses batch files. There is no database, shared queue, service account, web session, or multi-user permission model in the inspected implementation. This keeps local execution simple but means progress recovery, caching, audit history, and access separation must be supplied by the surrounding workflow.

6. Command Surface and Ownership

find

The find command performs the complete discovery waterfall for a person and organization domain. It collects domain context, scrapes public pages, generates conventional candidates, evaluates implemented signals, ranks the result, and formats evidence for review.

verify

The verify command evaluates a supplied candidate through syntax, disposable-domain, avatar, code-hosting, and related rules. Its score formula differs from discovery ranking, so its number should be interpreted within the verify command rather than compared as if all commands share one probability model.

scrape

The scrape command exposes public-site address collection without the full person-specific generation and ranking workflow. It is useful for understanding what the organization publishes directly.

batch

The batch command repeats supported discovery work across structured input rows. It shares the same network and heuristic limitations as individual execution; volume does not improve certainty and increases the importance of pacing, purpose controls, and review.

7. Five-Stage Discovery Pipeline

Stage 1 — Domain mail context

The program performs DNS lookups for mail exchange, sender-policy, and reporting-policy records. It also maps visible mail exchange patterns to eight recognized provider families. This provides organization-level context, not a person-level mailbox answer.

Stage 2 — Public website collection

The scraper checks two secure base variants and 14 conventional paths, allowing up to 28 serial page attempts. It extracts mail links, plain-text address forms, and a small set of common obfuscations, then removes known junk patterns.

Stage 3 — Candidate generation

When a person name is supplied, the pattern module creates eight common business-address arrangements. These are hypotheses based on naming convention, not observed addresses.

Stage 4 — Signal evaluation and filtering

Candidates pass syntax checks, a bundled disposable-domain screen, an avatar lookup, a public code-hosting search, and name-related comparison. Unsafe or irrelevant strings found in page text are filtered before ranking.

Stage 5 — Explainable ranking

The scorer assigns a base according to provenance and then applies explicit bonuses or penalties. The output contains the candidate, score, and available reasons so a reviewer can distinguish public observation from generated inference.

8. Input Normalization and Identity Assumptions

Names and domains enter through CLI arguments or batch rows. The program normalizes enough text to generate common patterns and construct network requests, but current domain canonicalization is

limited. Internationalized domains, unusual subdomains, display-name ambiguity, multi-part surnames, and organization aliases can produce misleading candidates.

A person name is not a unique identifier. Two people can share the same name, a public address can belong to a role rather than a person, and a domain can host several organizations. The tool should retain the operator's original inputs beside every result so a reviewer can understand the hypothesis that produced it.

9. DNS Intelligence

DNS inspection requests mail exchange records and evaluates sender-policy and reporting-policy presence. Provider-family detection recognizes eight implemented patterns. These signals help describe how a domain handles mail and whether common authentication records are published.

Mail exchange presence does not prove a specific mailbox. Sender-policy and reporting-policy presence do not prove correct enforcement or good deliverability. Provider identification may fail for custom gateways and can change when infrastructure changes. In the current implementation, DNS failures often collapse to empty or null results, so absence in output can mean not published, not resolved, timed out, or not recognized.

10. Public Website Scraping

The scraper constructs two secure base variants and checks 14 conventional paths such as organizational, team, contact, and policy locations. Requests are serial, use a browser-like user agent, have a 15-second timeout, and are subject to a process-global delay. A worst-case domain can therefore take significant time even before external candidate signals run.

Address extraction is regular-expression based over raw response text. It recognizes mail links, ordinary textual forms, and limited obfuscation. It then filters obvious resource names and irrelevant tokens. Because it does not build a full browser document or execute scripts, addresses rendered only by client-side code may be missed, while strings in scripts or metadata may be mistaken for visible contact information.

The helper named as a header-only status check currently performs a full GET request. This affects bandwidth, server load, and operator expectations. Redirect and final-destination validation also need careful interpretation because the current local CLI assumes an operator-supplied business domain rather than an adversarial multi-user request.

11. Pattern Generation

The generator emits eight conventional combinations derived from normalized first and last name components. The highest-ranked generated form is the first conventional pattern, while the remaining forms start with a lower base. This ordering reflects a heuristic preference, not organization-specific evidence.

Generated candidates must be labeled as inferred. They should not be merged with publicly observed addresses without provenance. If the website reveals a reliable organization-wide pattern, a reviewer may use that as stronger context, but the current generator does not mathematically prove that the same convention applies to every employee or role.

12. Syntax and Disposable-Domain Screening

Syntax validation filters strings that do not match the implemented address form. The bundled disposable-domain set contains 101 unique domains after normalization. A disposable match receives a substantial penalty in ranking.

Syntax only establishes that a string resembles a supported address. It does not prove mailbox existence. The disposable list is finite and time-sensitive, and a domain not on the list is not automatically trustworthy. Domain matching behavior also deserves care: substring-style comparisons can create false positives or false negatives where exact registrable-domain logic would be more appropriate.

13. External Public Signals

The avatar check derives the service lookup identifier from a normalized candidate using MD5. MD5 is used here because the public avatar service defines that lookup convention, not as password protection or secure hashing. The request discloses a deterministic representation of the candidate to the external service.

The code-hosting signal searches publicly accessible content using the raw candidate. A match can support the conclusion that the string has appeared publicly in a development context, but it does not prove current employment, mailbox control, or permission for outreach. Both integrations can fail, throttle, change behavior, or create privacy implications simply by receiving the lookup.

14. Discovery Scoring Model

Discovery ranking uses explicit integer rules. A publicly scraped candidate that also contains the supplied person-name signal begins at 85. Another public candidate at the organization domain begins at 65. The first generated pattern begins at 45, and other generated patterns begin at 35.

The scorer then applies implemented evidence adjustments: 20 points for an avatar signal, 10 for a public code-hosting signal, 5 for a name match, and a 50-point penalty for a disposable-domain match. The final value is clamped from 0 through 99.

This score is ordinal and explanatory. A score of 85 does not mean an 85 percent chance of correctness. The base values encode developer judgment about provenance, and the bonuses can correlate with each other. Scores should be compared alongside their reasons and source labels, not used as an automated outreach threshold.

15. Standalone Verification Model

The verify command begins with a separate base of 50, then adds 25 for an avatar signal and 15 for a public code-hosting signal, subject to its validation rules. Because this differs from discovery scoring, identical-looking numbers from find and verify have different derivations.

Verification here means evidence aggregation, not mailbox verification. It does not contact a mail server, send a confirmation message, or prove the candidate belongs to the named person. Client-facing output should use phrases such as “signals found” or “candidate confidence” rather than “verified mailbox.”

16. Ranking, Provenance, and Output Contract

Every useful result needs three distinct ideas: the candidate string, its provenance, and the reasons affecting its score. Provenance identifies whether the address was observed on a public page, generated from a naming pattern, or supplied directly for evaluation. Reasons identify signals, penalties, and name or domain relationships.

The output contract should remain stable enough for both humans and batch consumers. Empty findings should be explicit. Network failures should not masquerade as negative evidence. If a machine-readable output mode is consumed downstream, changes to field names, score meanings, or provenance labels are compatibility changes.

17. Batch Processing Lifecycle

Batch processing reads each structured input, validates required fields, performs the same discovery pipeline, and emits a result associated with the source row. The network delay is process-global, so execution is intentionally serial and can be slow across many domains.

There is no persistent checkpoint or durable job ledger. If the process ends mid-run, surrounding automation must determine which rows completed and whether replay is acceptable. Results can also change between runs as public websites, DNS, and external services change. A batch report should record execution time and tool revision to preserve interpretability.

18. Network Behavior and Operational Load

HTTP requests use a 15-second timeout and a browser-like user agent. The shared client enforces roughly half a second between process requests. Site scraping can attempt up to 28 pages for one domain, and later signal lookups add more calls. Therefore, “one lookup” can represent many external requests.

The implementation currently has no persistent response cache, exponential retry policy, provider quota coordinator, or circuit breaker. Broad exceptions frequently become absent results so the CLI remains usable, but that makes outages hard to distinguish from genuine no-signal outcomes. Responsible operation should use modest volumes and retain enough diagnostics to identify degraded sources without exposing researched personal data unnecessarily.

19. Module Ownership Map

- CLI parser and command handlers: argument contract, command routing, exit behavior, and presentation.
- Domain and name utilities: normalization used by all downstream modules.
- DNS module: mail exchange, policy-record, and provider-family context.
- Scraper: page selection, requests, extraction, obfuscation handling, and junk filtering.
- Pattern generator: eight deterministic person-name candidate hypotheses.
- Syntax and disposable checks: basic candidate admissibility and known temporary-domain penalty.
- Avatar signal module: deterministic public avatar lookup.
- Code-hosting signal module: public candidate appearance search.
- Scoring module: base provenance values, adjustments, clamp, and explanation.
- Batch module: row iteration and association between inputs and outputs.
- Test programs: deterministic expectations for core transformations and command behavior.

20. Data Flow and State Boundaries

Operator inputs flow into normalization. The domain flows to DNS and website modules. The name and domain flow to pattern generation. Publicly extracted and generated candidates flow through filtering and signal checks. Evidence flows into scoring. Ranked results flow to terminal or batch output.

There is no internal durable database. External services receive only the request material needed by each implemented lookup, but that material can still be sensitive because it represents a person-domain hypothesis. Results remain local unless the operator stores or transmits them. The code does not create consent or retention records, so those controls belong to the surrounding client process.

21. Privacy, Safety, and Responsible Use

DNS and public-site requests disclose the organization domain to network providers and destination servers. The avatar lookup discloses a deterministic MD5 representation of a full candidate. The code-hosting search discloses the raw candidate to that service. Operators must understand that even an unsuccessful lookup creates external request metadata.

Use should be limited to a defined legitimate purpose, the minimum necessary inputs, moderate volume, and human review. Public availability does not remove privacy obligations. A high score does not create consent. Suppression requests, do-not-contact choices, retention limits, data correction, and jurisdiction-specific outreach rules must be enforced outside the tool.

The CLI is not a hardened multi-user network service. If wrapped in an API, untrusted domains could cause server-side request-forgery and private-network access risks unless DNS resolution, redirect destinations, allowed ports, address ranges, and response sizes are constrained. Authentication, quotas, tenant isolation, audit trails, and deletion controls would also become required.

22. Security Boundaries

The inspected repository does not require embedded credentials for its core public-signal workflow. Configuration and logs should nevertheless avoid storing researched personal data longer than necessary. Terminal output, redirected files, shell history, and batch input are part of the security boundary because the application itself has no access-control layer.

Network responses are untrusted. Page size, redirects, encoding, malformed markup, and unexpected binary content can affect resource use and parsing. Domain inputs need strict canonicalization before any server-hosted deployment. Error messages should identify the failed source and stage without copying full response bodies or private environment paths.

23. Failure Behavior and Recovery

Invalid CLI arguments should fail before network activity. Syntax-invalid candidates should be rejected or clearly downgraded. DNS exceptions currently become empty or null context in many paths. Website request and parsing errors often yield no extracted addresses. External-signal failures generally behave like signal absence.

This fail-soft approach allows remaining stages to produce candidates, but it weakens evidence interpretation. The output should distinguish unavailable, attempted-with-no-match, and not-attempted states. Without that distinction, an outage can lower rankings as if it were meaningful negative evidence.

Batch interruption has no built-in checkpoint. Recovery is safe only when output association prevents completed rows from being duplicated or confused. Repeated public lookups can change results and add external exposure, so replay should be deliberate rather than automatic and unlimited.

24. Test Evidence — 2026-07-21

All four deterministic test programs passed. Together they contain 60 assertion statements covering core local behavior. This is strong evidence that the inspected implementation’s deterministic parsing, pattern, validation, scoring, and command expectations agree with its tests.

The test pass does not certify the current behavior of DNS providers, public websites, avatar services, code-hosting search, redirects, rate limits, or network error paths under real conditions. Those integrations were intentionally not called during this documentation review. The tests also do not measure address correctness against a labeled real-world data set.

25. Test Interpretation and Quality Boundaries

Deterministic tests are most useful for protecting explicit rules: eight patterns remain eight patterns, base scores and adjustments remain explainable, junk and disposable filters behave as expected, and command output retains its contract. A passing score algorithm test means the algorithm was implemented as specified; it does not validate the assumptions behind the chosen weights.

External integration tests should separate transport availability, parser compatibility, rate-limit behavior, redirect safety, and evidence semantics. Accuracy evaluation would require a consented, representative, time-bounded reference set and metrics that distinguish public observation, inferred pattern, mailbox existence, identity ownership, and deliverability. Those are different questions.

26. Operational Guidance

Run EmailFinder locally in a controlled environment with an explicit research purpose. Preserve the original input, execution time, tool version, candidate provenance, signal status, and human disposition if results enter a client workflow. Avoid logging complete candidate lists where a summary is sufficient.

Expect latency from serial page checks and external timeouts. A completed command can still contain degraded evidence when a provider was unavailable. Review diagnostics before comparing scores. Since no persistent cache or checkpoint exists, batch size should be chosen so interrupted work remains manageable.

27. Verified Limitations and Risk Register

- Scores are heuristic rankings, not calibrated probabilities.
- No SMTP, mailbox handshake, catch-all detection, or deliverability proof is implemented.
- Name and domain normalization does not cover every international or organizational form.
- Domain matching can be affected by substring logic and limited registrable-domain handling.
- Website extraction uses regular expressions over raw response text and does not execute browser scripts.
- Up to 28 serial site requests can make one discovery slow and externally visible.
- Broad error collapse can make provider failure look like absent evidence.
- Redirect and destination controls are insufficient for exposure as an untrusted public API.
- External avatar and code-hosting checks disclose candidate-derived data.
- The disposable-domain set is finite and becomes stale over time.
- There is no persistent cache, checkpoint, retry coordinator, or multi-user access-control model.
- High confidence does not establish consent, ownership, deliverability, or legal permission to contact.

28. Current Status Summary

EmailFinder is a complete local CLI for explainable public-signal research with a small dependency surface, four clear commands, and passing deterministic tests. Its strengths are transparency, no SMTP probing, explicit provenance, and an auditable score model. Its principal boundaries are heuristic certainty, external-service privacy exposure, ambiguous fail-soft network results, and the absence of hardened service controls.

This summary describes the current implementation. The documentation update does not alter discovery code, project data, or external systems.

29. Developer Handoff

A maintainer should trace one find command from argument parsing through normalization, DNS, each scraped path, pattern generation, syntax and disposable checks, external signals, scoring, sorting, and output. Then trace one network exception and one interrupted batch to understand where the current implementation intentionally loses status detail.

Changes should preserve provenance, explainable scoring, no-SMTP behavior, bounded network activity, stable output contracts, and clear separation between evidence and conclusion. If the CLI becomes a service, treat that as a new security architecture rather than a packaging change.

Appendix A. Representative Execution Traces

Discovery trace

Person and organization input → normalization → DNS context → bounded public-page attempts → public-address extraction → eight generated patterns → filtering → avatar and code-hosting signals → explainable score → ranked local output.

Verify trace

Supplied candidate → normalization and syntax → disposable-domain check → external public signals → verify-specific score → evidence summary with no mailbox guarantee.

Batch trace

Input row → validation → ordinary discovery pipeline → row-associated result → pacing → next row; interruption requires external identification of completed work.

Failure trace

External timeout or parsing failure → fail-soft empty signal → remaining stages continue → output must be reviewed for degraded evidence before interpretation.

Appendix B. Evidence Interpretation Matrix

- Public website observation: strongest implemented provenance, but can be stale, generic, or unrelated to the named person.
- Generated naming pattern: plausible hypothesis only; no direct observation.
- Mail exchange record: domain accepts some mail routing; says nothing about a specific mailbox.
- Sender or reporting policy: domain-level configuration context; not identity evidence.
- Avatar signal: candidate-derived external match; not proof of present ownership.
- Code-hosting signal: public occurrence of the candidate; not proof of employment or consent.
- Disposable-domain match: risk indicator based on a finite bundled list.
- Score: deterministic ordering under current rules; not a percentage of truth.

Appendix C. Developer Review Checklist

- Preserve the distinction between publicly observed, operator-supplied, and generated candidates.
- Keep score reasons visible and document every weight or formula change.
- Test domain normalization, internationalized inputs, redirects, and exact domain matching.
- Bound destinations, address ranges, ports, redirect depth, response size, and content types before any API exposure.
- Distinguish unavailable, no-match, and not-attempted external signals.
- Maintain explicit timeouts, pacing, and a truthful request method name.

- Keep external integrations replaceable and prevent provider failure from silently changing meaning.
- Protect batch input, terminal output, stored reports, and diagnostics as potentially personal data.
- Run deterministic tests and separately exercise controlled integration fixtures.
- Do not add SMTP probing or sending behavior without a new safety and compliance review.
- Version machine-readable output and record the scoring model with results.
- If adding persistence, define encryption, access, tenant isolation, retention, deletion, and audit ownership.

Appendix D. Client Review and Acceptance Checklist

- Confirm the research purpose, permitted input source, geography, and responsible-use policy before operation.
- Confirm reviewers understand that scores are not probabilities and results are not verified mailboxes.
- Review candidate provenance and reasons instead of approving outreach from a numeric threshold alone.
- Treat inferred patterns more cautiously than addresses visibly published by the organization.
- Confirm external avatar and code-hosting lookups are acceptable for the client's privacy policy.
- Define retention, access, correction, suppression, and deletion handling for inputs and outputs.
- Use human review to resolve namesakes, role addresses, stale pages, and organization-domain ambiguity.
- Confirm no output is interpreted as consent, legal permission, delivery assurance, or current employment proof.
- Record acceptance of the workflow and interpretation rules separately from acceptance of the software running successfully.

Developed by Zahidul Islam

Email discovery, evidence-aware scoring, responsible research, privacy boundaries, and maintainable CLI engineering.