

# Coding Protocol

February 13, 2026

## 1 Coding Protocol

**Coding unit and inputs.** The unit of coding is a paper. For each paper, the coder uses only (i) the title and (ii) the abstract for stage assignment; author-provided keywords are consulted only when available and only to confirm author-stated positioning. No label is assigned based on inferred intent. All labels must be selected from the finalized taxonomy label set (Appendix ?? / `taxonomy_labels.csv` in the replication package).

**Coding record.** Each paper corresponds to one row in the coding sheet (`labeled_corpus.csv`). The following fields are populated during coding: *paper\_id*, *year*, *title*, *abstract*, *keywords (optional)*, *primary\_label*, *secondary\_labels (optional)*, *technique\_tags (optional)*, *layer\_tag (optional)*, *vuln\_type (optional)*, *evidence\_span*, *matched\_cue*, *rule\_id (optional)*, and *notes (optional)*. Evidence spans are copied verbatim from the title/abstract.

**Step 0: Pre-check (label validity).** Before assigning labels, verify that the candidate label exists in the finalized label set. If the label is not present in the label set, it cannot be assigned; instead, select the closest parent category that is explicitly supported by the text and record the mismatch in *notes* for later inspection.

**Step 1: Stage assignment (title-first).** Stage assignment follows a title-first rule.

- Read the title and match explicit phrases to the stage definitions and cues in the codebook.
- Assign exactly one primary stage label (e.g., VM.VD, VM.VA, VM.VR, VM.FI, VM.SBI) *only if* the title provides unambiguous evidence.
- If the title is generic or does not indicate a stage (e.g., “A Framework for Security”), set *primary\_label* to *unclear* and record a short reason in *notes* (e.g., “title does not state whether the contribution is detection or assessment”).

**Step 2: Resolve unclear cases using the abstract.** This step is executed only when Step 1 yields *unclear*.

- Read the abstract and locate explicit objective statements (typical patterns include “we propose/present a tool/method to …”, “our approach aims to …”).
- Determine the stage based on what the paper claims as the main outcome: (i) *detection*—the outcome is a detector/scanner/predictor that flags vulnerabilities; (ii) *analysis/prioritization*—the outcome is severity/risk scoring, assessment, or ranking; (iii) *repair*—the outcome is patch generation or automated repair; (iv) *fix identification*—the outcome is locating fixes/fixing commits, advisory-to-fix mapping, or identifying silent fixes; (v) *bug identification*—the outcome is triaging or predicting security-related bug reports/issues.
- If the abstract still lacks explicit evidence for a stage, keep *primary\_label* as *unclear*. Do not force an assignment; add a brief explanation in *notes*.

**Step 3: Assign a specific subcategory under the stage (when explicitly supported).** After the stage is determined, a more specific label is assigned if the title/abstract explicitly supports it.

- Select the *most specific* label supported by the text (e.g., under detection: general-purpose vs. smart-contract vs. layer-specific; static vs. dynamic; learning-based vs. non-learning analysis; and representation type such as sequence / graph-to-sequence / graph; fuzzing for dynamic).
- If only the stage is explicit but not the subcategory, assign the stage-level label only (e.g., VM.VD) and leave finer labels unassigned.
- Do not “upgrade” to a fine-grained label unless the abstract/title contains direct signals (e.g., “GNN”, “message passing”, “coverage-guided fuzzing”, “CVSS ranking”, “advisory-to-commit mapping”).

**Step 4: Attribute tagging (technique, layer, vulnerability type).** Tags are recorded only when explicitly stated.

- *Technique tags:* Record author-stated descriptors of the approach (e.g., “GNN-based”, “Transformer”, “coverage-guided fuzzing”, “static analysis”). If a technique is not explicitly stated, leave the field empty.
- *Layer tag:* Assign a layer tag when the paper explicitly scopes its target (e.g., web/API applications; middleware/frameworks/libraries; network stacks/protocols; kernel/binary/compiler). If the scope is not stated but the abstract explicitly names the evaluated dataset or target artifact (e.g., kernel modules, binaries, network stacks, Android apps), map the layer based on that artifact and record the corresponding dataset/artifact phrase as evidence. If the evidence is ambiguous, leave the layer tag empty.
- *Vulnerability type:* Record the vulnerability type only when explicitly named in the title (e.g., “reentrancy”, “ReDoS”, “UAF”). If not named, leave the field empty even if it is implied.

**Step 5: Multi-stage handling (explicit multi-stage claims only).**

- If the title/abstract explicitly claims multiple VM stages, assign multiple labels.
- Select the primary label based on the dominant objective and contribution emphasized in the abstract; record other stages as *secondary\_labels*.
- If multi-stage is not explicitly claimed, do not assign multiple stages based on potential applicability.

**Step 6 : Representation tagging (model family, input structure, and graph encoding).** Representation tags are assigned only when the paper provides explicit evidence in the title/abstract (or an unambiguous model description).

- *Model family:* Tag the core learner as a **sequence model** (e.g., encoder-only, encoder-decoder, decoder-only) or a **graph model** (GNN/message passing). If the model type is not stated, leave the field empty.
- *Input structure:* Record whether the model input explicitly uses a **tree** (e.g., AST) or a **graph** (e.g., PDG/CFG/CPG/heterogeneous graphs). If the structure is only serialized into tokens and fed as a sequence, tag it as **sequence** rather than tree/graph.
- *Graph/tree encoding evidence:* If a tree/graph input is claimed, record whether the paper explicitly specifies **adjacency/edge encoding** (e.g., adjacency matrix, edge list, typed edges, edge-type matrix) together with node features/embeddings. If edge/adjacency encoding is not described, keep the tree/graph tag and leave this field empty.

**Step 7: Evidence logging (required for every assigned label).** For each assigned label (primary and secondary), record an auditable evidence trail:

- Copy the exact supporting span(s) from the title/abstract into *evidence\_span* (verbatim, short, and directly relevant).
- Record the matched cue in *matched\_cue* (the word/phrase in the text that triggered the label, aligned with codebook cues).
- If a tie-breaking rule is applied, record the rule identifier in *rule\_id* and summarize the rationale in *notes*.

**Disambiguation rules for overlapping labels (fine-grained overlap only).** Disambiguation is applied only when fine-grained labels overlap (e.g., repair vs. fix identification).

1. **D1 (repair):** Patch generation or code modification is treated as repair.

2. **D2 (fix identification):** Linking vulnerabilities to fixing commits, advisory-to-fix mapping, or locating security fixes is treated as fix identification.
3. **D3 (conservative):** If both signals appear or neither is explicit, keep only the parent label (or record the additional stage as secondary when explicitly claimed) and mark the case as *unclear* in *notes* for later inspection.

When needed, similar papers under the same top-level category are checked only to keep wording interpretation consistent; final labels still rely on explicit title/abstract evidence.

**Quality control checks.** After coding, a set of consistency checks is applied to the completed coding sheet to ensure that labels are valid, evidence is traceable, and formatting is consistent. These checks are performed at the corpus level and do not introduce new labels.

- **QC1 (exactly one primary stage):** Each paper must have exactly one *primary\_label*. If a paper is genuinely ambiguous after applying the protocol, *primary\_label* is set to *unclear* (rather than assigning multiple primary labels). Multi-stage is captured only via *secondary\_labels*.
- **QC2 (label validity):** Every *primary\_label* (except *unclear*) and each entry in *secondary\_labels* must match a valid *label\_id* in the finalized label set (*taxonomy\_labels.csv*). Any label not found in the label set is treated as invalid and must be replaced by the closest parent label that is explicitly supported by the text, with the adjustment documented in *notes*.
- **QC3 (evidence completeness):** For every assigned label (primary and secondary), *evidence\_span* must be non-empty and copied verbatim from the title and/or abstract. If no explicit evidence span can be identified, the label must be removed or replaced by a more conservative parent label, and the reason recorded in *notes*.
- **QC4 (cue traceability):** For each non-unclear label, *matched\_cue* must be provided and should correspond to an explicit word or phrase in the title/abstract that aligns with the codebook cues.
- **QC5 (disambiguation documentation):** If an overlap-disambiguation rule is used (e.g., repair vs. fix identification), the applied *rule\_id* (D1/D2/D3) must be recorded. The *notes* field should briefly state the conflicting signals and the explicit cue(s) that triggered the decision, with corresponding *evidence\_span* provided.
- **QC6 (secondary label formatting and scope):** When present, *secondary\_labels* must be formatted as a semicolon-separated list of valid *label\_ids* (e.g., VM.VD;VM.VA). Secondary labels are allowed only when multi-stage involvement is *explicitly claimed* in the title or abstract; otherwise, the field is left empty.

- **QC7 (conservative handling of fine-grained labels):** Fine-grained subcategory labels are assigned only when explicit textual evidence exists in the title/abstract (and keywords only when consulted for confirmation). If such evidence is missing, only the supported higher-level (parent) label is retained and finer labels remain empty. Uncertainty is documented in *notes* rather than resolved by inference.