

Policy coverage and workflow reuse

Experiment 1 addresses **RQ1: Baseline generation** and **RQ2: workflow reuse**. Each policy class has its own human-authored Policy Specification, Baseline Fact Design, and labeled samples. The parser interface, extractor skeleton, Rego driver template, prompt structure, runtime, and acceptance procedure are reused.

Table 1: Policy classes and the AST evidence required for their checks.

Policy class	Operation examined	Required AST evidence
CWE-321	Cryptographic key construction	Key-value origin and use in a cryptographic operation
CWE-89	JDBC execution	SQL value reaching an execution point
CWE-22	Filesystem operation	Path value reaching a filesystem operation
CWE-643	XPath evaluation	Expression value reaching XPath evaluation
CWE-117	Logging call	Value supplied to a logging function
CWE-284	Resource or database mutation	Sensitive operation and associated authorization

What this represents. The six classes require different program evidence. Successful use of the same generation and validation process across them demonstrates workflow reuse; it does not establish equal detection accuracy across the six classes.

Table 2: Coverage of seed expectations and post-Baseline evaluation patterns.

Measurement	Result	
Policy classes reproducing expected seed findings	6 / 6	
Post-Baseline evaluation patterns	22	Structural variations
Evaluation patterns already covered by the Baselines	9 / 22	
Evaluation patterns initially missed	13 / 22	
Initially missed patterns added by refinement	12 / 13	Refinement in policy design
Final evaluation-pattern coverage	21 / 22	CWE-284

What this represents. Baseline generation succeeded for all six policy classes, and accepted refinements added support for 12 previously missed evaluation patterns. Seed recovery is an acceptance condition. The breadth experiment does not provide corpus-level precision and recall for all six classes.

Source: the supplied manuscript’s Evaluation section. The blocks below reorganize its reported results. Arithmetic derived from the reported counts is explicitly identified.

Generation effort and manual implementation time

Table 3: Recorded pipeline effort by policy class. Time is in minutes.

	Class / scope	B	E	Att	Rev	Diag	LLM	Time
	CWE-89	6	2	2	8	31	108	28
	CWE-22	6	4	2	0	0	18	6
	CWE-643	6	4	3	4	4	66	12
	CWE-117	6	4	4	4	10	71	17
	CWE-284	7	4	4	2	7	61	11
	Five-class subtotal	31	18	15	18	52	324	74
	CWE-321 (larger campaign)	8	4	24	34	10	87	239
	Combined total	39	22	39	52	62	411	313

The five-class subtotal is calculated from the five non-CWE-321 rows. The CWE-321 row includes the later four-target refinement campaign from Experiment 2 and has a different scope. The combined total is an accounting total across these campaigns.

Table 4: How to read the generation-effort columns.

Column	What it counts
B	AST patterns supported by the accepted Baseline.
E	Post-Baseline evaluation patterns; some may already be supported.
Att	Candidate submissions across generated artifacts, as reported in the source.
Rev	Revision requests.
Diag	Failure-feedback diagnostic repair steps within revisions.
LLM	Total model requests recorded by the pipeline.
Time	Elapsed ATLAS generation-and-validation time, in minutes.

What this represents. Generation cost varies across policy classes even though the workflow is reused. B and E are separate pattern counts, not a success fraction. Att, Rev, and Diag describe different units of work and should not be added to reconstruct LLM requests. The CWE-321 campaign cannot be used as a like-for-like comparison of policy difficulty with the other five rows.

Table 5: Descriptive manual implementation time for each non-CWE-321 class.

	Activity	Approximate active time
	Inspect the AST	10 min
	Implement the Python AST extractor	50 min
	Implement the Rego policy checker	26 min
	Total per policy class	86 min

What this represents. One research-team member recorded about 86 minutes of active manual implementation per class; ATLAS took 6–28 minutes of elapsed generation and validation per class. These measure different kinds of time and do not establish a controlled productivity effect. Separately, authoring the CWE-321 Baseline Fact Design took 30 minutes.

Execution settings and the CWE-321 corpus

Experiment 2 addresses **RQ3: Baseline accuracy** and **RQ4: refinement**. Its primary measurement unit is one Java case. Pattern and cell counts describe structural coverage.

Table 6: Reported execution configuration.

	Setting
	Model and serving
	Workstation
	Refinement executions
	Generation settings
	Attempt budget
	Candidate acceptance
	Deterministic execution, Fact Design validation, expected-finding ch

What this represents. The three executions use the same reported generation settings but independent stochastic generations. Their outcomes describe observed variability under this configuration.

Table 7: Corpus composition and counting units.

	Scope	Violating	Safe	Total
	V1 within one pattern: identifier renaming	38	38	76
	V2 within one pattern: equivalent control construct	37	37	74
	One AST pattern (V1 + V2)	75	75	150
	Entire corpus: 12 AST patterns	900	900	1,800

Transformation-row totals and the 900/900 class totals are arithmetic consequences of the reported corpus composition.

Case. One Java example, counted once according to whether the accepted analyzer set produces the expected outcome.

AST pattern. One policy-relevant value-propagation or construction structure. There are 12 patterns, P01–P12.

Pattern–transformation cell. One AST pattern under V1 or V2. There are $12 \times 2 = 24$ cells. A cell is covered when its violating cases are detected; its safe cases separately test discrimination.

What this represents. The balanced corpus tests both detection and rejection of closely matched safe cases. Each safe case preserves the cryptographic constructor, surrounding structure, and nearby non-key literals, but obtains its key from a parameter or external input. Reporting every nearby literal would therefore produce false positives.

Evaluation scope. The corpus is author-constructed. All target-pattern cases participate in deterministic candidate validation and may supply failure evidence to later attempts. The results describe performance on this validation corpus; they do not measure performance on an untouched test set or production repositories.

Generation examples and structural coverage

Table 8: Baseline generation exposure and outcomes in the detailed run.

Patterns	Baseline generation exposure		
P01–P03	Three violating seed patterns		
P04–P08	Concrete examples not supplied	Additional structures using relations already handled	
P09	Not supplied	Method-reference path with aliasing and	
P10	Not supplied	Indirect construction	
P11	Not supplied		
P12	Not supplied	Method	

“Not supplied” refers to Baseline generation. P09–P12 each receive one violating example when their refinement begins. One matched safe control is used only for Baseline validation and is not a violating seed pattern.

What this represents. The Baseline is generated from P01–P03 and also detects P04–P08. The human-authored Policy Specification and Baseline Fact Design already name the broader provenance categories: `variable_chain`, `builder_chain`, `helper_return`, and `conditional_literal`. The extra coverage demonstrates reuse of supported AST relations within those specified categories.

Table 9: Refinement-example exposure and deterministic validation.

	Cases by role	Per target	Across P09–P12
Violating case supplied in the initial refinement prompt		1	4
Other violations absent from the initial refinement prompt		74	296
Matched safe cases absent as refinement examples		75	300
Cases used for target-pattern validation		150	600

Across-target counts are calculated by multiplying the per-target counts by four. They count distinct target cases, not repeated executions of those cases across candidate attempts.

What this represents. A refinement starts from one missed violation, but its acceptance depends on all 75 violating and 75 safe cases for the target pattern, together with regression checks. Absence from the initial prompt does not make the remaining cases an independent test set.

Accepted refinements are retained alongside the validated Baseline and earlier accepted refinements. Cumulative findings are the union of these analyzers’ findings. A rejected candidate leaves the accepted analyzer set unchanged.

Cumulative accuracy in the detailed run

The following results describe the **accepted analyzer set** after each decision. All rows are evaluated over the same 900 violating and 900 safe Java cases.

Table 10: Accepted coverage and case counts after each refinement decision.

	Accepted state / decision	Patterns / 12	Cells / 24	TP	FN	FP	Recall	Δ Recall
	Baseline: P01–P08	8	16	600	300	0	0.667	—
	P09 rejected; state unchanged	8	16	600	300	0	0.667	+0.000
	P10 accepted	9	18	675	225	0	0.750	+0.083
	P11 accepted	10	20	750	150	0	0.833	+0.083
	P12 accepted: final state	11	22	825	75	0	0.917	+0.083

TP: detected violations. FN: missed violations. FP: safe cases incorrectly reported. All accepted states have 900 true negatives (safe cases correctly left unreported), derived from 900 safe cases and zero FP. “Cells” replaces the source table’s misleading “n Samples” label.

What this represents. Each accepted refinement adds one AST pattern, two pattern–transformation cells, and 75 detected violations. Previously covered patterns remain covered, and no safe cases are reported. Rejecting P09 preserves the accepted results; its candidate findings are not included.

Table 11: Baseline and final accepted accuracy in the detailed run.

	Metric	Baseline	Final	Interpretation
	Precision	1.000	1.000	Fraction of reported cases that are violations.
	Recall	0.667	0.917	Fraction of all violations detected.
	F1	0.800	0.957	Harmonic mean of precision and recall.

Baseline F1 is calculated from the reported counts; the other values are reported in the manuscript. Values are rounded to three decimal places.

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad F_1 = \frac{2TP}{2TP + FP + FN}.$$

What this represents. The detailed run adds 225 detected violations, raising recall by $225/900 = 0.250$, or **25 percentage points**, while retaining precision 1.000. The remaining 75 missed violations all belong to P09. The aggregate gain is calculated from exact counts; individually rounded increments of 0.083 should not be summed to obtain it.

For supported patterns, the extractor can emit facts for both violating and safe occurrences. Safe facts identify external input and absence of hard-coded material; the Rego policy checker uses that provenance to suppress a policy finding. Zero false positives refers to findings, not to an absence of facts for safe code.

The rejected P09 candidate and variation across runs

Table 12: P09 candidate performance on its own 150 target-pattern cases.

	Measurement	Value	Meaning
	True positives	75	All P09 violations were detected.
	False negatives	0	No P09 violation was missed by the candidate.
	False positives	75	Every matched safe case was also reported.
	True negatives	0	No matched safe case was correctly left unreported.
	Precision	0.500	Only half of the reported target cases were violations.
	Recall	1.000	The candidate detected every target violation.
Acceptance decision		Rejected	Safe-case errors prevent retention of the candidate.

FN, TN, and recall are derived from the reported 75 violating cases, 75 safe cases, 75 TP, and 75 FP. These are candidate results on P09 only, not cumulative results for the accepted analyzer set.

What this represents. Perfect target recall was insufficient for acceptance because the candidate failed to distinguish hard-coded keys from runtime-derived keys. The manuscript reports that its generated Fact Design omitted provenance fields and AST references needed to distinguish the matched safe path. This diagnostic observation guides revision; it does not prove a unique root cause.

Table 13: Observed final outcomes across three refinement executions.

Execution	Accepted refinement targets	P09 decision	Patterns covered	Final recall
Run A	P09, P10, P11, P12	Accepted	12 / 12	1.000
Run B	P09, P10, P11, P12	Accepted	12 / 12	1.000
Detailed run	P10, P11, P12	Rejected	11 / 12	0.917

Pattern counts follow from the eight Baseline-supported patterns plus the accepted refinement targets. The source reports final recall for all three runs; the detailed candidate and cost breakdowns concern the detailed run only.

What this represents. Runs A and B produced acceptable P09 artifacts with sufficient distinguishing information. In the detailed run, rejecting the over-matching candidate preserved the previously accepted analyzer set and left final recall at 0.917. The observed final-recall range is 0.917–1.000.

Three executions are insufficient to estimate acceptance, rejection, or false-rejection rates. The artifacts were scored during their generation runs rather than replayed under one fixed toolchain, so runtime changes may also affect comparisons.

Refinement cost and what the results support

Table 14: Refinement cost in the detailed run. Time is in minutes.

	Target / decision	Stage-0	Rev	Diag	Rego	LLM	Time
	P09: rejected	11	9	6	9	29	84
	P10: accepted	7	25	4	18	50	132
	P11: accepted	3	0	0	1	4	10
	P12: accepted	3	0	0	1	4	13
	Total	24	34	10	29	87	239

Reading the counts. Stage-0 counts model requests in the pipeline’s Stage-0 category; Rev counts revision requests; Rego counts Rego-review requests. LLM is their sum:

$24 \text{ (Stage-0)} + 34 \text{ (Rev)} + 29 \text{ (Rego)} = 87 \text{ requests.}$

Diag is included within Rev and is not added again. The supplied results do not further define Stage-0’s internal activities, so its recorded label is retained here.

What this represents. Refinement cost varies sharply by target. P11 and P12 passed on their first generated artifacts in 10 and 13 minutes. P10 required 50 model requests and 132 minutes. The rejected P09 candidate consumed 29 requests and 84 minutes without adding accepted coverage. These observations describe this campaign; they do not establish intrinsic policy or pattern difficulty.

Table 15: How the reported evidence answers the research questions.

RQ	Question focus	
RQ1	Baseline generation	
RQ2	Workflow reuse	
RQ3	Baseline accuracy	On the 1,800-case CWE-321 corpus, the Baseline
RQ4	Refinement	Accepted refinements raised final recall to 1.000 in Runs A and B and 0.

What this represents. The experiments demonstrate generation and validation of policy-specific AST analysis pipelines, workflow reuse, and increased coverage of evaluated structural patterns through accepted refinements. The reported accuracy applies to the author-constructed validation corpus. It does not establish general Java data-flow capability, production accuracy, or accuracy across all six policy classes.