AI Data-Training and Labeling Infrastructure

Xaish AI is an applied research lab and data platform that curates, annotates and evaluates high-quality security data for training and testing machine learning and artificial intelligence models.

247
labeled examples across 8 security domains

100%
of the 144 code, cloud and payment examples carry a written rationale

366
live model calls in one evaluation of 2 frontier AI models

1
label corrected after an independent check disagreed, with the reason recorded

How a record is made

1 CURATE	Realistic, complete inputs: whole smart contracts rather than three-line fragments, real package manifests, real cloud policies. Every set is balanced between cases that should be flagged and cases that should not, and each deliberately includes **hard negatives**: safe inputs that look dangerous, which is where AI models over-react.
2 ANNOTATE	Each example gets a label and a **written rationale** that names the weakness (a CWE, CVE or scanner rule) or says exactly why the input is safe. The rationale is kept away from the model under test, so it cannot leak the answer.
3 VERIFY	Labels are checked against tools that run, not only against opinion: open-source security scanners such as Slither, Bandit and detect-secrets, and the OSV vulnerability database for package versions. A test re-checks every 'clean' package version against OSV and fails if one has gained a known vulnerability. When a check disagrees with a label, we find out which is wrong and record why.
4 EVALUATE	A harness runs any AI model or agent over the set and scores it: recall, precision, false-positive rate, F1, a confusion matrix and a verdict per example, with repeated runs to measure variance. It decides which model ships in production.

Why security data

Every AI lab now trains and tests models on code, and every company deploying an AI agent needs to know whether it can be trusted near production. General labeling work is moving to large crowds and synthetic data; the value is in expert judgment on hard, specialised cases. A security label is also unusually checkable: a vulnerable package version is in a public database or it is not, and a scanner either confirms a weakness or it does not. Xaish builds labels that can be verified, not only agreed on.

ANNOTATED RECORDS

Four records from the set

Each shows the input, the label, the rationale an expert wrote, how the label was checked, and what two AI models did with it in our September 2026 evaluation.

A real flaw

Code vulnerabilities (CWE) · V01-sqli-concat

INPUT	`def get_user(conn, name):`
	`  q = "SELECT * FROM users WHERE name = '" + name + "'"`
	`  return conn.execute(q).fetchall()`

LABEL	**vuln**
RATIONALE	CWE-89 SQL injection: query built by string concatenation of user input.
VERIFIED BY	CWE-89: user input concatenated into SQL. Paired with the safe version below.
MODELS	Claude Opus 4.7: **caught it** · Claude Opus 5: **caught it**

Its hard negative

Code vulnerabilities (CWE) · C01-sqli-parameterized

INPUT	`def get_user(conn, name):` `  return conn.execute("SELECT * FROM users WHERE name = %s", (name,)).fetchall()`
LABEL	**clean**
RATIONALE	Safe: parameterized query. Hard negative: superficially resembles V01.
VERIFIED BY	Same function, parameterized query. Paired with V01 so a model that matches on the word SELECT is caught out.
MODELS	Claude Opus 4.7: **correctly passed it** · Claude Opus 5: **correctly passed it**

A live-shaped secret

Leaked secrets · S01-aws-key

INPUT	`AWS_ACCESS_KEY_ID = "AKIA6O…"` `AWS_SECRET = "wJalr9…"`
LABEL	**vuln**
RATIONALE	Live-shaped AWS access key id + secret in source.
VERIFIED BY	Matches the AWS access-key format. Paired with Amazon's documented example key below; masked in this sample.
MODELS	Claude Opus 4.7: **caught it** · Claude Opus 5: **missed it**

A secret that is not one

Leaked secrets · C02-aws-example

INPUT	`AWS_ACCESS_KEY_ID = "AKIAIOSFODNN7EXAMPLE"`
LABEL	**clean**
RATIONALE	The AWS DOCUMENTATION example key. Every naive scanner flags it; it is not a real secret. HARD NEGATIVE.
VERIFIED BY	AKIAIOSFODNN7EXAMPLE is the example key printed in Amazon's own documentation.
MODELS	Claude Opus 4.7: **false alarm** · Claude Opus 5: **false alarm**

LABEL AUDIT

When the tool and the label disagree

A corrected label

Smart contracts · C08-event-emit

INPUT	`// SPDX-License-Identifier: MIT`
	`pragma solidity ^0.8.20;`
	
	`contract DepositBook {`
	`  mapping(address => uint256) public deposits;`
	`  uint256 public totalDeposits;`
	`  uint256 public depositCount;`
	
	`  event Deposited(address indexed who, uint256 amount, uint256 newBalance);`
	
	`  function deposit() external payable {`
	`  require(msg.value > 0, "zero deposit");`
	`  deposits[msg.sender] += msg.value;`
	`  totalDeposits += msg.value;`
	`  depositCount += 1;`
	`  emit Deposited(msg.sender, msg.value, deposits[msg.sender]);`
	`  }`
	
	`  function balanceOf(address who) external view returns (uint256) {`
	`  return deposits[who];`
	`  }`
	
	`  function averageDeposit() external view returns (uint256) {`
	`  if (depositCount == 0) {`
	`  return 0;`
	`  }`
	`  return totalDeposits / depositCount;`
	`  }`
	`}`

LABEL	**vuln**
RATIONALE	Locked ether (SWC-131 adjacent): deposit() is payable and credits deposits[], but the contract has NO withdrawal function, so every wei sent is unrecoverable by anyone including the depositor. Relabelled 2026-10-02; it was 'clean' with the rationale 'Safe: deposit + event, checked math, no external call', which enumerated the absent hazards and missed the absent exit. Slither's locked-ether detector flags it independently, so the scanner was right and the label was wrong - the eval had been scoring a miss as a success.
VERIFIED BY	Slither's locked-ether detector, run independently of the label.
HISTORY	Labeled **clean** until 2 October 2026, now **vuln**. Until then the evaluation counted the scanner's correct flag as a false alarm.

Evaluating a defence, not only a model

The same method measures the Xaish Runtime Shield: 65 labeled attack requests in 8 families and 38 ordinary requests, some written to look suspicious. It blocked 58 of 65 attacks (89%) and 1 of 38 ordinary requests (2.6%). Every miss is listed in the published scorecard.

EVALUATION

Two frontier models, 7 tasks, one decision

On 18 September 2026 we ran Claude Opus 4.7 and Claude Opus 5 over the same labeled cases, 366 live calls in all. The newer model was worse on 4 of 7 tasks; most seriously, it missed the live-shaped AWS key on page 2. We kept Claude Opus 4.7 in production. Without labeled data, the upgrade would have shipped on reputation.

Task	Examples	Claude Opus 4.7 recall · false alarms	Claude Opus 5 recall · false alarms
Code vulnerabilities (CWE)	20	100% · 20%	100% · 40%
Python security audit	20	100% · 20%	100% · 60%
Leaked secrets	20	100% · 60%	90% · 40%
Vulnerable dependencies	20	100% · 44%	100% · 44%
Cloud identity policies	20	100% · 30%	100% · 60%
Smart contracts	20	100% · 100%	100% · 100%
Payment fraud	24	92% · 0%	92% · 0%

Red: worse than the model we kept. False alarms are the share of safe examples the model wrongly flagged.

DATA CARD

What the set contains today

Domain	Examples	Should flag	Should pass
Code vulnerabilities (CWE)	20	10	10
Python security audit	20	10	10
Leaked secrets	20	10	10
Vulnerable dependencies	20	10	10
Cloud identity policies	20	10	10
Smart contracts	20	11	9
Payment fraud	24	12	12
Web attack traffic	103	65	38
Total	**247**	**138**	**109**

FORMAT	One JSON object per line: id, label, payload (the input exactly as the model sees it) and rationale (never shown to the model).
ANNOTATORS	The Xaish founding engineers. Each label was checked by a tool that runs, where one exists.
SIZE	247 examples. This is a proof of method, built to choose and gate the models behind 146 production agents. It is not yet a volume dataset and is not yet sold to third parties.
NEXT	Grow each domain with security specialists as annotators, add agent-trajectory and fix-quality labels (was the patch correct, did the test plan hold), and offer the sets and the evaluation harness to teams training or deploying AI on code.

samuel@xaish.com

Source files: evals/datasets/*.jsonl, evals/results/_sweep_opus47 and _sweep_opus5, evals/results/runtime_shield.json. Generated by scripts/investor_docs/make_data_sample.py.