

Vyper on Ethereum: an on-chain data review

Independent measurement, offered as input to the End-to-End Formally Verified Vyper Compiler initiative. No affiliation with the Vyper team, the Vyper Foundation, TheDAO Security Fund or Giveth. This document contains measurements only. It contains no view on whether the initiative should be funded, and no recommendations. Chain data read 28 September to 8 October 2026 via Dune SQL over Ethereum tables, with compiler versions read from Etherscan verified sources. The SQL behind every figure is published alongside this report.

1. How this was measured, in brief

Vyper emits no compiler version tag in deployed bytecode, so contracts are identified here by their dispatcher preamble. Against 1,462 contracts whose language was labelled independently from verified source, the method runs at **100% precision and 100% recall**. The appendix has the detail.

Ethereum mainnet only, so L2 deployments are excluded. ERC-20 holdings only, so native ether is excluded and value figures understate. Balance figures are snapshots that drift a few percent a week on token prices. Contracts nobody ever verified sit outside both the detector and the test of it, so counts remain floors.

2. What Vyper secures

Measure	Value
Vyper contracts ever deployed on Ethereum	9,829
Holding a positive ERC-20 balance	2,038
USD value of those holdings	about 3.28bn

3. Who holds it

Six deployer addresses hold 80% of all Vyper-secured value, out of 378 addresses holding any. Consolidating those deployers by the organisation that controls them, across the top 40 deployers (3.23bn USD, which is 98.6% of the base):

Organisation	USD held	Share
Curve	2,330,045,601	72.2%
Yield Basis	881,473,564	27.3%
Unattributed small holders	10,327,951	0.3%
Yearn	6,802,120	0.2%
Keep3r Network	517,339	0.02%
LayerZero (scrvUSD bridge)	58,893	0.002%

Curve's share spans twelve distinct product lines: the crvUSD controller and peg keepers, the stableswap NG factory, veCRV, twocrypto, 3pool, tricrypto, LlamaLend V2 and its vaults, the crvUSD flash lender, FraxBP and the renBTC pool.

On Yield Basis specifically. It is the second-largest constituency by a wide margin, holding roughly 881m USD across two deployer addresses, the second being a factory created by the first in September 2025. It compiles on 0.4.3, ahead of most of the Curve estate. The initiative page lists Curve and ChainSecurity as prospective early users and does not mention it.

Caveat on the unit: a deployer address is not an organisation. Attribution above is by resolving names and creation lineage on Etherscan and Dune labels, not by proving common control.

4. Whether a verified mode can reach that value

A verified mode binds at deployment. Live contracts cannot be recompiled in place, so first-year coverage is capped by how much value sits in contracts deployed within the window. Measured:

Window	Share of held value
Contracts deployed in the last 365 days	14.4%
Contracts deployed 365 to 730 days ago	34.0%
Combined, under two years old	48.4%

The second window uses the same construction and the same as-of balances with the deployment window shifted back a year, so the figure is not an artefact of one window. The trailing-year number is a floor on what a year's deployments eventually hold: a contract deployed eighteen months ago has had longer to accumulate deposits than one deployed three months ago.

5. Deployment volume and trend

In the 90 days to 8 October 2026, Ethereum saw 4,755,282 contract creations. Of those, 4,030,208 were EIP-1167 minimal-proxy clones, 457,165 were Solidity-style and **506 were Vyper-style**. Against Solidity

alone, ignoring clones, Vyper is 506 of 457,671, or **0.111%** of hand-written deployment.

By quarter since 2023 Q2, Vyper deployment has stayed between **302 and 1,108** contracts for thirteen consecutive quarters. Absolute volume is up about 27% year on year, while Solidity grew faster over the same span, so Vyper's share of hand-written deployment fell by roughly a third.

6. Compiler provenance is not observable on chain

Searching every Ethereum contract creation over seven days for any Vyper compiler tag, in CBOR-encoded or plain ASCII form, returned **zero matches across 155,587 creations**, against **35,712** creations carrying a Solidity version tag.

Inspecting the trailing bytes of confirmed Vyper contracts on 0.3.10 and 0.4.3 shows a hash-shaped trailer and no version string. [vyperlang/vyper#3471](#) moved the CBOR version tag out of runtime code, and [issue #3549](#) asking to revert it has been open since 3 August 2023.

7. Which compiler versions production actually uses

From the verified sample described in the appendix, n=57:

Compiler	Contracts	Share
vyper 0.3.10	28	49.1%
vyper 0.4.3	19	33.3%
vyper 0.4.2	4	7.0%
vyper 0.3.9	3	5.3%
vyper 0.3.7, 0.4.0, 0.4.1	1 each	5.3%
vyper 0.5.0	0	0%

The 0.3 line is about half of new deployment and the 0.4 line about 44%. By value rather than count the picture differs: among the six largest deployers, five sit on 0.3.10, 0.3.7 or 0.2.4, and the sixth, Yield Basis, is on 0.4.3.

8. Contracts still running bytecode from the 2023-affected compilers

A compiler fix reaches new output only, and deployed bytecode is immutable. So the 2023 patch did not reach contracts the affected versions had already emitted. That population was measured.

78 Vyper contracts deployed before the August 2023 fix still hold more than 1,000 USD, together about 1.15bn USD. Of those:

- **4 contracts**, holding **about 4.33m USD** combined, were compiled with 0.2.15, 0.2.16 or 0.3.0 and share a reentrancy lock key across two or more functions. That is the structural condition for the 2023

defect.

- **None of the four holds an asset that returns control to the caller mid-transfer**, which is the second ingredient the 2023 exploit required. The largest, 98.8% of that 4.33m, is a stablecoin pair with no ether leg and no transfer hooks. The remaining three are voting escrows locking ordinary ERC-20 governance tokens.
- **6 contracts**, holding about **1.89m USD**, have no published source, so their compiler version cannot be established by any public means. That is 0.058% of the base, and smaller than the identified affected set rather than larger.

This is a source and asset review, not an audit, and not a statement that any contract is safe. Specific addresses are deliberately not listed here; they are available to the Vyper team or to any affected project on request.

9. Corrections welcome

The unverified population is the weakest point: recall is measured against source-labelled contracts, so a Vyper contract nobody verified is invisible to both the detector and the test of it. Anyone with a better identification method, or a counterexample, is more useful to this analysis than agreement.

Appendix. Identification method and its validation

Why a bytecode pattern is needed. Vyper emits no compiler version tag in deployed bytecode; Solidity does. So a Vyper contract cannot be identified, or its version read, from bytecode alone.

The patterns. A contract counts as Vyper when its deployed runtime code begins with any of:

Pattern	Covers
5f3560e01c	0.3.10 and later, PUSH0 dispatcher
60003560e01c	0.4.x without PUSH0
6003361161000c57	0.3.9 era
600436101561000d57	0.2.x to 0.3.x
341561	0.2.7 and 0.2.8, CALLVALUE guard ahead of the dispatcher
fe71	ERC-5202 blueprints, initcode containers from create_from_blueprint

Solidity runtime code always begins with the free-memory-pointer setup 6080604052, so none of these collides with it.

Blueprints are counted separately. They are initcode containers, hold no balance by design, and cannot appear in a value figure. Two variants are therefore correct for different questions: value questions use the first five patterns, deployment counts use all six.

Validation, against 1,462 contracts whose language was labelled from verified source independently of their bytecode:

	Result
Precision, against 526 Blockscout-verified Solidity contracts	0 false positives, 100%
Precision, two independent random samples of detected contracts on Etherscan	57 of 57 Vyper, both times, 100%
Recall, all six patterns, against 936 Blockscout-verified Vyper contracts	936/936, 100%
Recall, five value-bearing patterns, against 866 non-blueprint contracts	866/866, 100%

Recall cannot be measured from the detector's own output, since sampling what it catches tests only precision. The ground truth comes from Blockscout's `?filter=vyper`, which returns contracts whose source language was recorded at verification from submitted source. 1,100 were collected and 936 had retrievable runtime bytecode.

The figures in this report use the four-pattern variant, without `341561` or `fe71`, which measures **833/866 = 96.2%** recall over value-bearing contracts. Its misses are 0.2.7 and 0.2.8 contracts. Every value figure in sections 2, 3 and 4 is therefore a floor that understates slightly, and none can fall under the wider patterns. Section 5's deployment counts understate by more, since blueprints were 7.5% of the ground truth and the `341561` family another 3.5%; Vyper's 0.111% share of hand-written deployment would rise toward roughly 0.12%.

The limit that remains. Recall is measured over Vyper contracts verified on Blockscout. A Vyper contract nobody ever verified is outside the ground truth and outside the detector, and that population is real: 6 of the 78 pre-fix contracts in section 8 have no published source. The claim is 100% recall against 936 source-labelled contracts, with the unverified population unquantified.