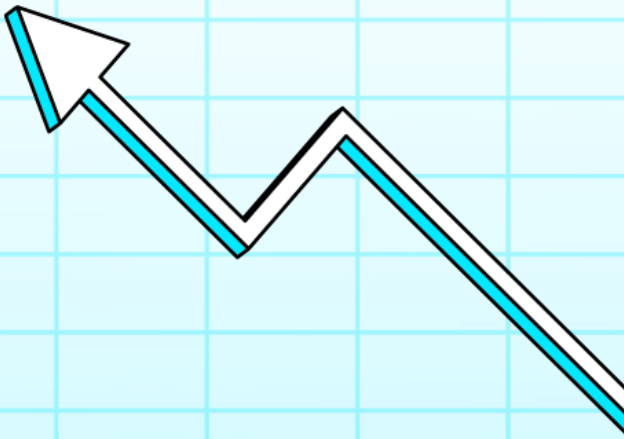
shieldify



Bankroll Vault Token (VLT)

SECURITY REVIEW

Date: 1 June 2026

CONTENTS

1. About Shieldify

Positioned as the first hybrid Web3 Security company, Shieldify shakes things up with a unique subscription-based auditing model that entitles the customer to unlimited audits within its duration, as well as top-notch service quality thanks to a disruptive 6-layered security approach. The company works with very well-established researchers in the space and has secured multiple millions in TVL across protocols, also can audit codebases written in Solidity, Vyper, Rust, Cairo, Move and Go.

Learn more about us at shieldify.org.

2. Disclaimer

This security review does not guarantee bulletproof protection against a hack or exploit. Smart contracts are a novel technological feat with many known and unknown risks. The protocol, which this report is intended for, indemnifies Shieldify Security against any responsibility for any misbehavior, bugs, or exploits affecting the audited code during any part of the project's life cycle. It is also pivotal to acknowledge that modifications made to the audited code, including fixes for the issues described in this report, may introduce new problems and necessitate additional auditing.

3. About Bankroll Vault Token (VLT)

Overview

`VaultToken` is a small, single-purpose, fixed-supply ERC20 with a one-shot `bootstrap()` function that seeds a Uniswap V2 pool and effectively locks the LP tokens by minting them to the contract's own address. There is no `mint()` function, no pause, no blocklist, no upgrade path, no selfdestruct, and no delegatecall. The deployer (the owner) has exactly one privilege - calling `bootstrap()` - and loses even that after the first invocation.

Executive Summary

The VaultToken (VLT) contract was deployed in June 2020 and bootstrapped 13 blocks later. It is immutable, fully bootstrapped, and operating exactly as designed. The bootstrap() function — the only privileged function in the contract — was called once and can never be called again.

- **VLT liquidity is rug-proof. 99.997848%** of the Uniswap V2 LP tokens are held by the VLT token contract itself and cannot be moved by anyone — including the owner, who holds **0 LP**. The protocol contains no code path capable of withdrawing that liquidity.
- **~$713K of liquidity (178.18 WETH + 900,445 VLT) is permanently locked** behind the contract as a liquidity floor (value as of snapshot; ETH/USD = $2,001.09).
- **All Uniswap 0.30% trading fees accrued over ~6 years are also locked**, because the LP they accrue to is frozen — they are unredeemable by any party.
- **3.84% of total supply (69,058.642933535960580372 VLT) is permanently frozen** at the contract address with no function able to move it; it can never circulate.
- The earlier supply-conservation observation is **unobservable on the live, already-bootstrapped contract** and has been reduced to a one-line note.

Supply Distribution

Bucket	Amount (VLT)	% of Supply	Raw (wei)	Class
In Uniswap V2 pool	900,445.459473	50.0247%	900445459473359157676844	Snapshot
Locked at contract	69,058.642933	3.8366%	69058642933535960580372	Permanent
Owner wallet	74,935.394359	4.1631%	74935394359393555445813	Snapshot
Other holders (circulating)	755,560.503233	41.9756%	755560503233711326296971	Snapshot
Total	**1,800,000.000000**	**100.00%**	1800000000000000000000000	—

All four buckets verified to sum to exactly 1,800,000 VLT (1800000000000000000000000 wei)

Note: Supply is monotonically non-increasing. Only `burn()` can reduce it. No `mint()` function exists. Supply can never exceed 1,800,000 VLT.

VLT Permanently Locked At Contract (3.84%)

```
Amount locked:    69,058.642933535960580372 VLT
                  (69058642933535960580372 wei)
Percentage:       3.8366% of total supply
Status:           Permanently immovable — verified
```

Trading Fee Implications

Uniswap V2 charges 0.30% on every swap. Fees accrue proportionally to LP token holders by increasing the value backing each LP token over time.

Learn more: docs

4. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1 Impact

- **High** – results in a significant risk for the protocol's overall well-being. Affects all or most users

- **Medium** – results in a non-critical risk for the protocol affects all or only a subset of users, but is still unacceptable
- **Low** – losses will be limited but bearable, and covers vectors similar to griefing attacks that can be easily repaired

4.2 Likelihood

- **High** – almost certain to happen and highly lucrative for execution by malicious actors
- **Medium** – still relatively likely, although only conditionally possible
- **Low** – requires a unique set of circumstances and poses non-lucrative cost-of-execution to rewards ratio for the actor

5. Security Review Summary

This is an on-chain verification of the deployed, immutable VLT contract and its Uniswap V2 liquidity.

The security review lasted 2 days, with a total of 32 hours dedicated by the Shieldify team.

Overall, the code is well-written. The audit report flagged one Informational recommendation regarding the ERC-20 supply conservation invariant not holding while the contract is in an unbootstrapped state.

The Bankroll team has done a great job and provided exceptional support to the Shieldify researchers.

5.1 Rug-Proof Determination — TRUE (on liquidity)

5.1.1 LP-Ownership Proof

LP metric (pair 0x966053Ca4fca049173eb1F27E4cb168CCb794534)	**Value**	**Raw (wei)**
LP totalSupply()	11,230.824900787623519053	11230824900787623519053
LP held by **token contract** 0x6b785a0322126826d8226d77e173d75DAfb84d11	11,230.583243981587618849	11230583243981587618849
% of all LP held by the contract	**99.997848%**	—
LP held by **owner** 0xcB71EB21F53a2F4de0F26dc90518Df10Be13D1EC	**0**	0
LP held by all other addresses (removable)	0.241656806035900204 (0.002152%)	241656806035900204
of which MINIMUM_LIQUIDITY burned to 0x0000000000000000000000000000000000000000	0.000000000000001000	1000

Derivation of the lock percentage: 112305832439815876188 49 / 112308249007876235190 53 = 0.99997848 **99.997848%**.

5.1.2 Why the locked LP can never move (code-level)

- The one-shot `bootstrap()` function (src/Contract.sol:208) seeds the pool via router. addLiquidityETH(...) with the to parameter set to `address(this)` (src/VaultToken.sol:247'). The freshly minted LP tokens are therefore credited to the **token contract itself**.
- The contract holds **no reference to the pair / LP-token and makes no call to it**. The only external liquidity call in the deployed runtime bytecode is `addLiquidityETH` to the router — selector 0 xf305d719, which appears exactly once. No `removeLiquidity*` selector is present (`0xbaa2abde`, `0x02751cec`, `0x2195995c`, `0xded9382a` all absent — verified against the deployed bytecode). The contract is therefore structurally incapable of moving its own LP balance.
- `bootstrap()` is guarded by `require(isBootStrapped == false, ...)` (src/VaultToken.sol:211) and sets isBootStrapped = true (src/VaultToken.sol:252). There is no path that resets the flag, so the function can never run again.

5.1.3 The owner cannot pull liquidity either

The owner address holds **0 LP tokens**. Even the sole privileged function (bootstrap()) is spent. There is no owner-only liquidity-withdrawal path. The team is therefore structurally incapable of removing pool liquidity.

5.1.4 Honest caveat (residual external LP)

0.002152% of the LP supply (0.241656806035900204 LP; 1,000 wei of which is the permanently burned Uniswap MINIMUM_LIQUIDITY) is held by ordinary external liquidity providers. They can remove their own share at will. This is immaterial and is not a rug vector against the protocol.

5.1.5 Scope of the claim

"Rug-proof" here means **pool liquidity cannot be removed**. It does **not** claim immunity to ordinary market price movement: the owner's VLT (Section 6/7) and other holders' VLT can be sold into the pool like any token, which moves price. That is normal market activity, not a liquidity rug.

Determination: 99.998% of pool liquidity is permanently unwithdrawable by any party. **VLT is rug-proof on liquidity.**

5.2 Protocol Summary

Project Name	Bankroll Vault Token (VLT)
Repository	vlt
Type of Project	ERC-20, Store-of-Value Token
Security Review Timeline	2 days
Review Commit Hash	0x6b785a0322126826d8226d77e173d75dafb84d11

5.3 Scope

The following smart contracts were in the scope of the security review:

File	nSLOC
VaultToken.sol	108
Total	**108**

6. Findings Summary

The following issues have been identified, sorted by their severity:

- **Info** issues: 1

ID	Title	Severity	Status
[I-01]	ERC-20 Supply Conservation Invariant Does Not Hold While the Contract Is Not Bootstrapped	Info	Acknowledged

7. Findings

[I-01] ERC-20 Supply Conservation Invariant Does Not Hold While the Contract Is Not Bootstrapped

Severity

Informational Risk

Description

The core ERC-20 conservation invariant requires that the reported total supply always equals the sum of all account balances:

```
totalSupply() == balanceOf(a)  for every address a
```

`VaultToken` does **not** satisfy this invariant while the contract is not yet bootstrapped. `_totalSupply` is hard-initialized to `1_800_000e18` at its declaration, but **no `balances[]` entry is ever written until `bootstrap()` executes**. There is no constructor mint, no premint to the deployer, and no `_mint()`-style accounting that pairs a supply with a balance. The only assignment to any balance happens inside `bootstrap()`.

Consequently, for the entire window between deployment and a successful `bootstrap()` call:

- `totalSupply()` returns `1_800_000e18`.
- `balanceOf(a)` returns `0` for **every** address, including the owner and the contract itself.
- `balanceOf == 0 1_800_000e18 == totalSupply`.

The two supply sources also disagree during this window: `totalSupply()` reports `1.8M` while the event-reconstructed supply (sum of all `Transfer(address(0), ...)` mint events) is still `0`.

The invariant is **restored** the instant `bootstrap()` succeeds: that call writes `balances[owner]` and `balances[token]`, emits the matching mint events, sets `isBootStrapped = true`, and from then on `balanceOf == totalSupply()` holds permanently (the only other supply-mutating path, `burn()`, decrements supply and balance together).

Location of Affected Code

File: src/VaultToken.sol

```
// src/VaultToken.sol:194 — supply initialized at declaration, with no
   matching balance
uint256 _totalSupply = 1800000 * (10 ** 18); // 1.8 million supply

// src/VaultToken.sol:258-259 — view returns the phantom supply
   unconditionally
function totalSupply() public override view returns (uint256) {
    return _totalSupply;
}

// src/VaultToken.sol:262-263 — every balance is the default 0 until
   bootstrap() writes them
function balanceOf(address player) public override view returns (uint256)
    {
    return balances[player];
}

// src/VaultToken.sol:220-225 — the FIRST and ONLY place balances are
   assigned + mint events emitted
balances[owner] = _totalSupply * 89 / 100;
balances[token] = _totalSupply.sub(balances[owner]);
emit Transfer(address(0), owner, balances[owner]);
emit Transfer(address(0), token, balances[token]);

// src/VaultToken.sol:251-252 — successful bootstrap closes the not-
   bootstrapped window
isBootStrapped = true;
```

Note also that `burn()` (`src/Contract.sol:335-337`) decrements both `_totalSupply` and `balances[msg.sender]` together — i.e. the contract's only supply-mutating path elsewhere keeps the invariant intact. The declaration-time initialization is the lone place where supply is created without a corresponding balance, which is precisely what breaks the invariant in the not-bootstrapped state.

Impact

Information because the deployed instance The deployed instance `0x6b785a0322126826d8226d77e173d75dafb84d11` (Ethereum mainnet, created at block `10255188`, ~June 2020) is **already bootstrapped**, so this invariant break is not observable in the live contract. The invariant is satisfied at deployment and remains satisfied after the one-time bootstrap, so no external party can observe a state where it is violated.

Recommendation

The deployed contract is **immutable and already bootstrapped**, so no on-chain code change is possible or necessary — the live instance now satisfies the invariant. The recommendation is therefore **documentation, not a code fix**:

Team Response

Acknowledged.

shieldify

Thank you!