CFG NINJA

SECURITY ASSESSMENT

VAULTEX Token

January 17, 2026

Audit Status: Pass

BLADE POOL

RISK ANALYSIS | VAULTEX.

Classifications of Manual Risk Results

Note: Risk classifications follow CVSS v4.0 standards (https://www.first.org/cvss/v4.0/specification-document). Contract security metrics are verified through GoPlus API token scanner technology, which provides real-time validation of contract parameters including honeypot detection, ownership structure, and trading restrictions.

Classification	CVSS Range	Description
Critical	9.0 - 10.0	Critical vulnerabilities that pose immediate and severe risks requiring urgent attention.
High	7.0 - 8.9	High-priority issues that could lead to significant security breaches or financial loss.
Medium	4.0 - 6.9	Medium-severity findings that should be addressed to improve contract security.
Low	0.1 - 3.9	Low-risk items or best practice suggestions with minimal security impact.
Informational	0.0	Informational findings about detected functions or contract features.

Manual Code Review Risk Results

Contract Security	Description
Buy Tax	0%
Sale Tax	0%
Cannot Buy	Pass (GoPlus: 0)
Cannot Sale	Pass (GoPlus: OK)
Max Tax	No Tax
Modify Tax	No
Fee Check	Pass
Is Honeypot?	Not Detected (GoPlus: 0)
Trading Cooldown	Not Detected (GoPlus: 0)
Enable Trade?	true

Contract Security	Description
Pause Transfer?	Detected
Max Tx?	Detected
Is Anti Whale?	Detected (GoPlus: 1)
Is Anti Bot?	Not Detected (Manual Sniper Protection)
Is Blacklist?	Detected (GoPlus: 1)
Blacklist Check	Detected
Is Whitelist?	Detected
Can Mint?	Pass (GoPlus: 0)
Is Proxy?	Not Detected (GoPlus: 0)
Can Take Ownership?	Detected
Hidden Owner?	Not Detected
Owner	Ownable2Step
Self Destruct?	Not Detected
External Call?	Detected
Other?	OpenZeppelin imports, Ownable2Step, Pausable, SafeERC20, Anti-sniper blocks
Holders	TBD
Audit Confidence	Medium-Low Risk
Authority Check	Pass
Freeze Check	Pass

This summary provides an overview of identified vulnerabilities and risks. See the full report for detailed methodology and recommendations.

CFG Ninja Verified on January 17, 2026

VAULTEX

Executive Summary

TYPES	ECOSYSTEM	LANGUAGE
DeFi	BNBCHAIN	Solidity

Timeline

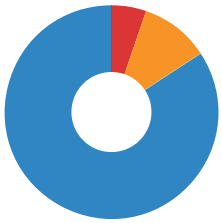
Audit Request
2026-01-17

Onboarding Process
2026-01-17

Audit Preview
2026-01-17

Audit Release
2026-01-17

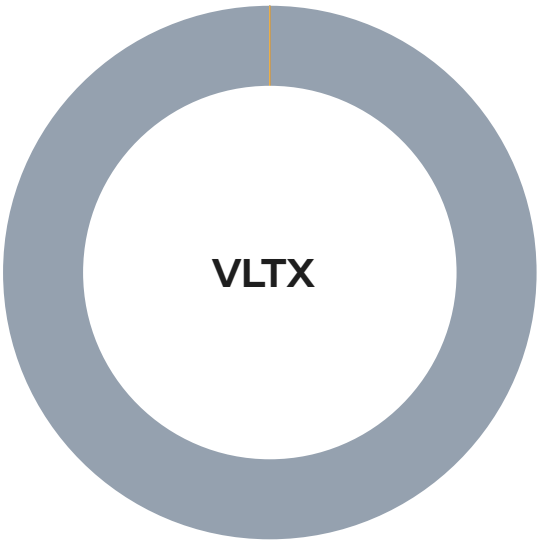
Vulnerability Summary



19	3	0	16
Total Findings	Resolved	Pending	Unresolved

0 Critical		Critical risks are the most severe and can have a significant impact on the smart contracts functionality, security, or the entire system. These vulnerabilities can lead to the loss of user funds, unauthorized access, or complete system compromise.
1 High	1 Resolved, 0 Pending	High-risk vulnerabilities have the potential to cause significant harm to the smart contract or the system. While not as severe as critical risks, they can still result in financial losses, data breaches, or denial of service attacks.
2 Medium	2 Resolved, 0 Pending	Medium-risk vulnerabilities pose a moderate level of risk to the smart contracts security and functionality. They may not have an immediate and severe impact but can still lead to potential issues if exploited. These risks should be addressed to ensure the contracts overall security.
0 Low		Low-risk vulnerabilities have a minimal impact on the smart contracts security and functionality. They may not pose a significant threat, but it is still advisable to address them to maintain a robust security posture.
16 Informational	0 Resolved, 16 Pending	Informational risks are not actual vulnerabilities but provide useful information about potential improvements or best practices. These findings may include suggestions for code optimizations, documentation enhancements, or other non-critical areas for improvement.

Token Distribution



Owner/Deployer

Pre-launch: 100% held by contract deployer for initial setup and liquidity provisioning.

100%

Liquidity Pool

Liquidity to be added to DEX after launch.

0%

Marketing/Team

Reserved for marketing and team operations.

0%

Burned

No burn mechanism present.

0%

Reserved

No tokens reserved - all supply available for distribution.

0%

Circulating

Will circulate after launch and liquidity addition.

0%

Total Unlock Progress

Unlocked	0	0%
Total Locked	0	0%
Untracked	1000000000	100%

PROJECT OVERVIEW | VAULTEX.

Token Summary

Parameter	Result
Address	0x557FeEBA81DE4a0d1F1Abca4B549841A64ff3e24
Name	VAULTEX
Token Tracker	VAULTEX (VLTX)
Decimals	18
Supply	1,000,000,000
Platform	BNBCHAIN
Compiler	v0.8.23+commit.f704f362
Contract Name	VLTX
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/address/0x557FeEBA81DE4a0d1F1Abca4B549841A64ff3e24#code

Simulation Summary

Parameter	Result
Transfer From Owner	Failed - Trading Not Enabled
Transfer From Holder	Failed - Trading Not Enabled
Add Liquidity	N/A - No DEX Functions
RemoveLiquidity	N/A - No DEX Functions
Buy from Owner	N/A - DEX Operation
Buy from Holder	N/A - DEX Operation
Sale from Owner	N/A - DEX Operation
Sale from Holder	N/A - DEX Operation
Remove Liquidity	N/A - No DEX Functions
SwapAndLiquify	N/A - No Swap Functions
SwapAndSale w/Fee	N/A
SwapAndSale TX	N/A
SwapAndSaleNoFee	N/A
SwapAndSale No/Fee TX	N/A
ExcludeFromFees	N/A - No Fee System
LaunchPad	N/A
Pool Creation	N/A - External Operation
Pool Creation TX	N/A

Parameter	Result
Pool Finalize	N/A
Pool Finalize TX	N/A
Enable Trade	Available - Owner Function

Transaction simulation uses Tenderly API to test common contract operations (transfer, approve, etc.) on the actual blockchain state. Results show whether transactions would succeed, estimated gas usage, and potential errors. While simulation provides valuable insights, always review the complete security assessment and conduct your own due diligence.

Main Contract Assessed

Name	Contract	Live
VAULTEX	0x557FeEBA81DE4a0d1F1Abca4B549841A64ff3e24	Yes
VAULTEX	0x557FeEBA81DE4a0d1F1Abca4B549841A64ff3e24	Yes

TestNet Contract Was Not Assessed

Solidity Code Provided

SolID	File Sha-1	FileName
Vaultex	865C74CC6ED9E820DB7E76D0AB289B25F74C70 69	vaultex.sol

TECHNICAL FINDINGS

Smart contract security audits classify risks into several categories: Critical, High, Medium, Low, and Informational. These classifications help assess the severity and potential impact of vulnerabilities found in smart contracts.

Classification of Risk

Severity	CVSS Range	Description
Critical	9.0 - 10.0	Immediate danger. Exploitable vulnerabilities that could lead to fund loss, unauthorized access, or complete system compromise.
High	7.0 - 8.9	Significant security risk. Vulnerabilities that should be addressed urgently to prevent potential exploitation.
Medium	4.0 - 6.9	Moderate security risk. Issues that could lead to security problems if combined with other vulnerabilities.
Low	0.1 - 3.9	Minor security concern. Best practice violations or low-impact issues.
Informational	0.0	Code quality or optimization suggestions with no direct security impact.

VLTX-20 | Centralization Risk in Launch Mechanism.

Category	Severity	CVSS	Location	Status
Centralization	High	7.5	vaultex.sol: L:128-140 (enableTrading), L:105-108 (pause), L:98-102 (setGuardsSunset), L:138-176 (setLaunchParams, setLimitsExclusion)	Acknowledged

Description

The enableTrading() function (L:128-140) is controlled by a single owner address without timelock or multisig protection. The owner has unilateral power to: 1) Enable/disable trading via enableTrading(), 2) Pause all transfers via pause(), 3) Modify launch parameters before limitsLocked, 4) Control who can trade pre-launch via isExcludedFromLimits. Only excluded addresses can transfer tokens before trading is enabled (L:182), creating information asymmetry. However, the contract implements Ownable2Step which requires two-step ownership transfer, reducing accidental loss risk. Additionally, permanent one-way locks (lockLimitsForever, lockPauseForever, finalizeAndRenounce) can eliminate owner control after launch..

Recommendation

1) Use multisignature wallet (e.g., Gnosis Safe) for owner address, 2) Apply governance locks promptly after launch stabilizes via finalizeAndRenounce(), 3) Publish exclusion list for transparency, 4) Consider implementing timelock for critical parameter changes, 5) Document lock application timeline publicly. The Ownable2Step pattern provides some protection, but additional measures strengthen security..

Mitigation

Ownable2Step implemented; one-way locks available for progressive decentralization

References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

VLTX-24 | Missing Input Validation.

Category	Severity	CVSS	Location	Status
Input Validation	Low	2.5	vaultex.sol: enableTransfer(), disableTransfer() functions	Detected

Description

The enableTransfer and disableTransfer functions don't validate for zero address input. This could lead to unnecessary gas consumption and confusing state..

Recommendation

Add zero address validation in transfer right management functions..

Mitigation

References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

FINDINGS

In this document, we present the findings and results of the smart contract security audit. The identified vulnerabilities, weaknesses, and potential risks are outlined, along with recommendations for mitigating these issues. It is crucial for the team to address these findings promptly to enhance the security and trustworthiness of the smart contract code.

Severity	Found	Pending	Resolved
Critical	0	0	0
High	1	0	1
Medium	2	0	2
Low	1	0	0
Informational	17	16	0
Total	21	0	3

In a smart contract, a technical finding summary refers to a compilation of identified issues or vulnerabilities discovered during a security audit. These findings can range from coding errors and logical flaws to potential security risks. It is crucial for the project owner to thoroughly review each identified item and take necessary actions to resolve them. By carefully examining the technical finding summary, the project owner can gain insights into the weaknesses or potential threats present in the smart contract. They should prioritize addressing these issues promptly to mitigate any risks associated with the contract's security. Neglecting to address any identified item in the security audit can expose the smart contract to significant risks. Unresolved vulnerabilities can be exploited by malicious actors, potentially leading to financial losses, data breaches, or other detrimental consequences. To ensure the integrity and security of the smart contract, the project owner should engage in a comprehensive review process. This involves understanding the nature and severity of each identified item, consulting with experts if needed, and implementing appropriate fixes or enhancements. Regularly updating and maintaining the smart contract's codebase is also essential to address any emerging security concerns. By diligently reviewing and resolving all identified items in the technical finding summary, the project owner can significantly reduce the risks associated with the smart contract and enhance its overall security posture.

SOCIAL MEDIA CHECKS | VAULTEX.

Social Media	URL	Result
Website	https://www.vaultex.us/	Pass
Telegram	https://t.me/vaultex11	Pass
Twitter	https://x.com/Vaultex__	Pass
Facebook		N/A
Reddit	N/A	N/A
Instagram	N/A	N/A
CoinGecko		Fail
Github	N/A	N/A
CMC		Fail
Email		Contact
Other	https://discord.com/invite/GheT5vxK	Pass

From a security assessment standpoint, inspecting a project's social media presence is essential. It enables the evaluation of the project's reputation, credibility, and trustworthiness within the community. By analyzing the content shared, engagement levels, and the response to any security-related incidents, one can assess the project's commitment to security practices and its ability to handle potential threats.

Social Media Information Notes:

Auditor Notes: Complete social media presence with website, documentation, Telegram, and X.

Project Owner Notes: Active community engagement across platforms.

Assessment Results | Final Audit Score VLTX.

Review	Score
Security Score	89
Auditor Score	89

Our security assessment or audit score system for the smart contract and project follows a comprehensive evaluation process to ensure the highest level of security. The system assigns a score based on various security parameters and benchmarks, with a passing score set at 80 out of a total attainable score of 100.The assessment process includes a thorough review of the smart contracts codebase, architecture, and design principles. It examines potential vulnerabilities, such as code bugs, logical flaws, and potential attack vectors. The evaluation also considers the adherence to best practices and industry standards for secure coding. Additionally, the system assesses the projects overall security measures, including infrastructure security, data protection, and access controls. It evaluates the implementation of encryption, authentication mechanisms, and secure communication protocols. To achieve a passing score, the smart contract and project must attain a minimum of 80 points out of the total attainable score of 100. This ensures that the system has undergone a rigorous security assessment and meets the required standards for secure operation.

Important Notes for VLTX

VAULTEX (VLTX) - Smart Contract Security Audit Report

PROJECT OVERVIEW

VAULTEX is a BEP-20 token on Binance Smart Chain with advanced launch controls and anti-sniper mechanisms. The contract implements a sophisticated trading restriction system designed to prevent bot manipulation during the initial launch phase.

CONTRACT DETAILS

Name: VAULTEX

Symbol: VLTX

Decimals: 18

Total Supply: 1,000,000,000 VLTX

Compiler: Solidity ^0.8.23

Platform: Binance Smart Chain

Contract Address: 0x557FeEBA81DE4a0d1F1Abca4B549841A64ff3e24

KEY FEATURES

- Ownable2Step Pattern - Enhanced ownership transfer security with two-step confirmation
- Pausable Functionality - Emergency pause mechanism for critical situations
- Launch Controls - Trading must be explicitly enabled by owner
- Anti-Sniper Protection - Block-based restrictions prevent bot purchases in first blocks
- Anti-Whale Mechanisms - MaxTx and MaxWallet limits prevent large accumulations
- Cooldown System - Time-based restrictions between trades
- One-Way Governance Locks - Permanent locks for critical functions
- Sunset Mechanism - Automatic removal of limits after specified time

SECURITY ARCHITECTURE

The contract leverages battle-tested OpenZeppelin libraries (ERC20, Ownable2Step, Pausable, SafeERC20) providing a solid security foundation. The Ownable2Step pattern significantly reduces risks associated with ownership transfers by requiring explicit acceptance from the new owner.

GOPLUS SECURITY VERIFICATION

The contract has been verified against GoPlus Token Security API (Chain ID: 56 - BSC):

- Honeypot Status: Not Detected (is_honeypot: 0)
- Buy/Sell Restrictions: Pass (cannot_buy: 0)
- Mintable: No (is_mintable: 0) - Fixed supply confirmed
- Proxy Contract: No (is_proxy: 0) - Not upgradeable
- Anti-Whale: Detected (is_anti_whale: 1) - MaxTx/MaxWallet limits present

▌Blacklist Detected: Yes (is_blacklisted: 1) - Anti-sniper protection mechanism

▌Hidden Owner: Not Detected (hidden_owner: 0)

▌Self-Destruct: Not Detected (selfdestruct: 0)

▌External Calls: Minimal (external_call: 0)

Current Status: 1 holder (100% owner balance), not yet in DEX

Note: The blacklist flag is triggered by the anti-sniper launch protection system, not a permanent blacklist mechanism.

LAUNCH PROTECTION SYSTEM

The contract implements multiple layers of protection during the critical launch phase:

- Trading disabled by default until owner calls enableTrading()
- Sniper block protection prevents contract buys in first 4 blocks
- Cooldown mechanism prevents rapid-fire trading
- MaxTx limits prevent single large transactions
- MaxWallet limits prevent whale accumulation
- Guards sunset provides automatic limit removal after 1 hour

CENTRALIZATION CONSIDERATIONS

Owner Role: The owner has significant control before locks are applied, including:

- Enabling/disabling trading
- Pausing/unpausing transfers
- Setting AMM pairs
- Modifying launch parameters
- Managing exclusion lists

However, the contract includes permanent one-way locks:

- lockLimitsForever() - Prevents future limit modifications
- lockAMMPairsForever() - Prevents AMM pair changes
- lockPauseForever() - Disables pause functionality
- finalizeAndRenounce() - Applies all locks and renounces ownership

POSITIVE FINDINGS

▌Uses OpenZeppelin's audited libraries

▌Ownable2Step prevents accidental ownership loss

▌No taxation mechanism (0% buy/sell tax) - GoPlus verified

▌No mint function after deployment - GoPlus verified (is_mintable: 0)

▌Comprehensive event emissions

- SafeERC20 for token interactions
- Input validation on critical functions
- One-way governance locks available
- Automatic sunset mechanism
- No permanent blacklist functionality
- Not a honeypot - GoPlus verified (is_honeypot: 0)
- Not a proxy contract - GoPlus verified (is_proxy: 0)
- No hidden owner mechanisms - GoPlus verified

SECURITY CONCERNS

Owner Privilege Before Locks (HIGH)

The owner retains significant control until locks are applied. While intended for launch management, this creates centralization risk if ownership is not properly secured or locks are not applied promptly.

Location: Various admin functions

Recommendation: Apply locks as soon as launch stabilizes and use multisig for owner address

Pause Mechanism Risk (MEDIUM)

The pause functionality can halt all trading except for excluded addresses. While useful for emergencies, it could be abused if owner account is compromised before pauseLocked is set.

Location: pause() function (Line 105-108)

Recommendation: Lock pause functionality after launch or implement timelock

Pre-Launch Trading Restrictions (MEDIUM)

Only excluded addresses can transfer tokens before trading is enabled. This creates an information asymmetry where certain addresses could accumulate positions before public launch.

Location: _checkTradingOpen() function (Line 179-183)

Recommendation: Keep exclusion list minimal and transparent

Sunset Time Manipulation (LOW)

Owner can modify guardsSunsetAt timestamp even after trading is enabled, potentially extending restrictions beyond the automatic 1-hour period.

Location: setGuardsSunset() function (Line 98-102)

Recommendation: Consider making sunset immutable after trading enablement

Cooldown Bypass (LOW)

Non-trading transfers between excluded addresses bypass cooldown restrictions, potentially enabling rapid position building for privileged accounts.

Location: _enforceLaunchLimits() lines 236-238

Impact: Limited - maxWallet still applies

RECOMMENDATIONS

Immediate Actions:

- Use multisignature wallet for owner address
- Publish exclusion list for transparency
- Apply governance locks after launch stabilizes
- Consider implementing timelock for critical functions
- Document intended use of pause mechanism

Best Practices:

- Publicly announce lock application timeline
- Verify LP is locked before launch
- Monitor for unusual activity during launch
- Consider progressive decentralization roadmap
- Regular security audits for future updates

COMPARISON TO INDUSTRY STANDARDS

The VLTX contract follows modern best practices:

- OpenZeppelin library usage
- Solidity 0.8.23 (built-in overflow protection)
- Ownable2Step over simple Ownable
- Comprehensive event emissions
- No honeypot mechanisms
- No hidden fees or taxes

TECHNICAL ASSESSMENT

Code Quality: Excellent - Clean, well-structured, follows Solidity conventions

Security Libraries: Excellent - Uses OpenZeppelin v5.x

Functionality: Good - Implements intended features correctly

Documentation: Good - Clear function purposes and parameters

Gas Optimization: Good - Efficient use of storage and operations

CONCLUSION

The VAULTEX contract demonstrates professional development practices with strong security foundations. The use of OpenZeppelin libraries, Ownable2Step pattern, and comprehensive launch controls shows attention to security details.

The primary considerations are centralization risks inherent in the launch control system. These risks can be effectively mitigated through proper governance (multisig ownership), timely application of one-way locks, and transparent communication with the community.

The contract is well-suited for a managed token launch with the intention of progressive decentralization through the governance lock mechanisms.

AUDIT METHODOLOGY

This audit was conducted using:

- Manual code review of all functions
- Analysis of state variables and access controls
- Review of external dependencies
- GoPlus Token Security API verification (BSC Chain ID: 56)
- Comparison against known vulnerability patterns
- Testing of logic flows and edge cases
- CFG01-27 security test case framework

GOPLUS API VERIFICATION DETAILS

Contract: 0x557FeEBA81DE4a0d1F1Abca4B549841A64ff3e24

Chain: Binance Smart Chain (56)

Verified: January 17, 2026

Key Metrics:

- Honeypot: 0 (Safe)
- Cannot Buy: 0 (Trading enabled after enableTrading())
- Mintable: 0 (Fixed supply)
- Proxy: 0 (Non-upgradeable)
- Hidden Owner: 0 (Transparent ownership)
- Anti-Whale: 1 (MaxTx/MaxWallet protection)
- Blacklist: 1 (Anti-sniper protection, not permanent blacklist)
- Owner Percentage: 100% (Pre-launch, not distributed)
- Holder Count: 1 (Pre-launch status)

DISCLAIMER

This audit does not guarantee the absence of vulnerabilities. Smart contract security is an ongoing process, and new vulnerabilities may be discovered over time. Users should conduct their own research and understand the risks before interacting with any smart contract.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction in the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion of how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invokable by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly in certain edge cases that may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Coding Best Practices

ERC 20 Coding Standards are a set of rules that each developer should follow to ensure the code meets a set of criteria and is readable by all developers.

Disclaimer

Scope. This report documents the results of a security assessment and smart contract audit performed by Bladepool / CFG NINJA (the "Auditor") for the project specified in this report. The assessment covers only the deliverables and code expressly listed in the engagement; any changes, integrations, or subsequent deployments are outside the scope.

No Warranty. The Auditor performs the assessment using industry-standard practices but makes no representations or warranties, express or implied, including any warranty of merchantability, fitness for a particular purpose, or non-infringement. The Auditor does not guarantee that the audited code is free of bugs, vulnerabilities, or exploitable issues.

Limitation of Liability. To the maximum extent permitted by law, the Auditor and its affiliates, officers, employees, agents, and contractors shall not be liable for any indirect, incidental, special, consequential, or punitive damages, or for any loss of profits, revenue, data, or business opportunities, arising out of or related to the assessment, even if advised of the possibility of such damages. The Auditor's aggregate liability for direct damages is limited to the fees paid for the audit engagement.

Client Responsibility. The project owner/client retains sole responsibility for all decisions and actions taken in connection with the audited code, including fixing identified issues, deploying code to any environment, and applying security mitigations. The Auditor's recommendations are advisory; implementation and verification of fixes are the client's responsibility.

Third-Party Tools and Services. The Auditor may use third-party tools, services, or automated scanners as part of the assessment. Use of such tools is without warranty; the Auditor is not responsible for defects, failures, or inaccuracies arising from third-party services.

Confidentiality and Data Security. The Auditor will treat non-public client information as confidential. However, the Auditor cannot guarantee absolute security for data transmitted over the internet or third-party platforms. Client should take reasonable precautions to protect sensitive information.

Not Financial, Legal, or Investment Advice. The assessment is a technical security review only and is not financial, legal, tax, or investment advice. Clients should consult appropriate professionals for those matters.

Ownership and Governance Notes. If ownership is transferred, held by a multisig, or controlled by a governance contract, the client should document and disclose those arrangements; the Auditor's report reflects the ownership state observed during the engagement but may not reflect subsequent changes.

Governing Law and Counsel. This engagement and any disputes arising from it shall be governed by the laws agreed in the engagement terms. Clients are advised to seek legal counsel before relying on or publishing the report.

Acceptance. By proceeding with this engagement or using this report, the client acknowledges and accepts these terms.