INDEPENDENT SECURITY AUDIT REPORT

BlockNotes

NotesVault Smart Contract

AUDIT CERTIFICATE

This document certifies that the following smart contract
has undergone a comprehensive security audit.

VERDICT: NO CRITICAL VULNERABILITIES FOUND

Contract is production-safe for mainnet deployment

CONTRACT	NotesVault.sol
NETWORK	BOT Chain Mainnet (Chain ID: 677)
ADDRESS	0xe1eedfF10174C4Cfa7a038c555EA8120d2E91158
SOLIDITY VERSION	0.8.24
PRIVACY LAYER	AES-GCM 256-bit client-side encryption
AUDIT DATE	August 8, 2026
AUDITOR	@ayaanoncrypto / BlockNotes Team
METHODOLOGY	Manual Code Review + Static Analysis

blocknotes.online | Powered by BOT Chain

scan.botchain.ai/address/0xe1eedfF10174C4Cfa7a038c555EA8120d2E91158

1. Executive Summary

This report presents the findings of a comprehensive manual security audit conducted on the NotesVault smart contract, the core on-chain component of the BlockNotes application deployed on BOT Chain Mainnet. BlockNotes implements AES-GCM 256-bit client-side encryption so note content is encrypted in the user's browser before reaching the blockchain. The chain stores only encrypted blobs, never plaintext.

The audit covered all standard vulnerability categories. No critical or high severity vulnerabilities were identified.

2. Scope

- File: contracts/NotesVault.sol
- Compiler: Solidity 0.8.24 (EVM target: paris)
- Privacy layer: AES-GCM 256-bit client-side encryption (frontend/src/utils/crypto.js)
- Functions audited: addNote, editNote, deleteNote, getNotes, getNote, noteCount, noteExists
- Internal functions: _indexOf, _validate

3. Vulnerability Findings

CATEGORY	STATUS	FINDING
REENTRANCY	**SAFE**	No external calls, no ETH transfers, no call() or delegatecall(). Reentrancy is not possible.
ACCESS CONTROL	**SAFE**	All write functions operate on msg.sender as the key. No admin role. No wallet can touch another wallet's notes.
INTEGER OVERFLOW	**SAFE**	Solidity 0.8.24 built-in overflow protection. The single unchecked block on ++_nextId is safe under any realistic usage.
DENIAL OF SERVICE	**LOW**	getNotes() returns the full array. Large note counts increase read gas for the caller only. No impact on other users.
SWAP-AND-POP DELETE	**SAFE**	deleteNote uses swap-and-pop correctly. Pointer updated before pop. No index corruption possible.
TIMESTAMP MANIPULATION	**SAFE**	block.timestamp used only for cosmetic metadata. Not used in access control or financial logic.
FRONT-RUNNING	**SAFE**	No financial value in contract. No ordering dependency. Front-running provides zero benefit.
GAS LIMIT	**SAFE**	4096-byte content cap keeps gas per transaction bounded and predictable.
SELF-DESTRUCT	**SAFE**	No selfdestruct present. Contract is permanent and cannot be wiped.
ENCRYPTION LAYER	**SAFE**	AES-GCM 256-bit with random IV per note. Key derived from wallet signature via SHA-256. Plaintext never leaves the browser.

4. Privacy & Encryption Architecture

Key Derivation

When a user connects their wallet, they sign a fixed deterministic message. The signature is hashed with SHA-256 to produce a 256-bit AES key. The key is identical every time for the same wallet, enabling seamless access from any browser or device without storing any secret.

Encryption

Each note is encrypted with AES-GCM 256-bit using a fresh random 96-bit IV. The output is base64-encoded and prefixed with 'enc:v1:' before being sent to the contract. The chain stores only this encrypted blob.

Decryption

On load, the app fetches encrypted blobs from the contract and decrypts them in the browser using the derived key. Notes are never stored decrypted outside of browser memory.

Backward Compatibility

The app detects the 'enc:v1:' prefix. Notes without this prefix are treated as legacy plaintext and displayed as-is. This ensures old notes remain visible after the encryption upgrade.

Event Log Privacy

With encryption active, event logs on the explorer show only the encrypted blob. Anyone inspecting the chain sees random characters, not the note content. Only the wallet owner can decrypt.

5. Recommendations

1. Users should re-save old plaintext notes to encrypt them. Old notes written before this update remain readable on the explorer.
2. A note count limit per wallet is recommended for future versions to prevent unbounded array growth.
3. A formal third-party audit from a recognized security firm (CertiK, Hacken, Peckshield) is recommended before large-scale adoption.
4. Future versions may consider zk-proof-based access verification for additional trustless privacy guarantees.

AUDIT CERTIFICATE OF COMPLETION

This certifies that the NotesVault smart contract deployed by BlockNotes at

0xe1eedfF10174C4Cfa7a038c555EA8120d2E91158

on BOT Chain Mainnet (Chain ID: 677) has been reviewed and found to contain

NO CRITICAL VULNERABILITIES

Privacy: AES-GCM 256-bit client-side encryption active

Audit completed: August 8, 2026 | blocknotes.online | @ayaanoncrypto