QuillAudits

AUDIT REPORT

September 2025

For

Hedge DAO

Table of Content

Executive Summary

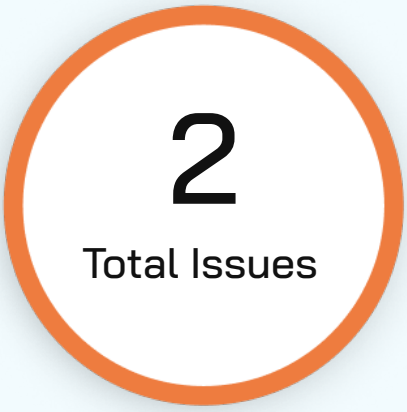
Project Name	HedgeX DAO
Protocol Type	Token
Project URL	https://hedgex.io/
Overview	HedgeX DAO is an ERC20 token contract that implements a deflationary mechanism through burn taxes on AMM trades (buys/sells on DEXs like Uniswap). When users trade through marked automated market maker pairs, a configurable percentage (default 0.1%) of tokens are burned from the transaction, reducing total supply over time. The contract includes a supply floor mechanism that stops burning once the total supply reaches a minimum threshold, preventing over-deflation. It combines multiple OpenZeppelin extensions including voting capabilities (ERC20Votes), snapshots for governance, capped supply, and standard burn functions, all controlled by an owner (intended to be a multisig wallet). The contract also tracks unique holder counts and allows the owner to exclude certain addresses from burn taxes, making it suitable for DAO governance with deflationary tokenomics.
Audit Scope	The scope of this Audit was to analyze the HedgeX Smart Contracts for quality, security, and correctness.
Source Code link	https://github.com/blokchain-ai-bu/Hedgex_Dao_Smart_Contracts_Quill
Branch	Main
Contracts in Scope	contracts/HedgexDao.sol
Commit Hash	ea85f54c0e805039a566855a5e4a607e5af6dc5f
Language	Solidity
Blockchain	EVM
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	27th August 2025 - 29th August 2025

Updated Code Received	1st September 2025
Review 2	4th September 2025
Fixed In	95d1b353a479dc1bf4f2709ed71e59b8c618a5ce

Verify the Authenticity of Report on QuillAudits Leaderboard:

https://www.quillaudits.com/leaderboard

Number of Issues per Severity



Critical	0 (0.0%)
High	0 (0.0%)
Medium	2 (100%)
Low	0 (0.0%)
Informational	0 (0.0%)

Severity

Issues	Critical	High	Medium	Low	Informational
Open	0	0	0	0	0
Acknowledged	0	0	0	0	0
Partially Resolved	0	0	0	0	0
Resolved	0	0	**2**	0	0

Summary of Issues

Issue No.	Issue Title	Severity	Status
1	Lack of slippage on _transfer can lead to users paying more tax than they expected	Medium	Resolved
2	Rounding leads to not paying burn taxes	Medium	Resolved

Checked Vulnerabilities

- Access Management
- Arbitrary write to storage
- Centralization of control
- Ether theft
- Improper or missing events
- Logical issues and flaws
- Arithmetic Computations Correctness
- Race conditions/front running
- SWC Registry
- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC's conformance
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls

- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Multiple Sends
- Using suicide
- Using delegatecall
- Upgradeable safety
- Using throw
- Using inline assembly
- Style guide violation
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code
- Use of best practices
- Code documentation and comments, match logic and expected behavior
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper
- Implementation of ERC standards
- Efficient use of gas
- Code is safe from re-entrancy and other vulnerabilities

The following techniques, methods, and tools were used to review all the smart contracts:

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms Used for Audit

Remix IDE, Foundry, Solhint, Mythril, Slither, Solidity Static Analysis.

Types of Severity

Every issue in this report has been assigned to a severity level. There are five levels of severity, and each of them has been explained below.

Critical: Immediate and Catastrophic Impact

Critical issues are the ones that an attacker could exploit with relative ease, potentially leading to an immediate and complete loss of user funds, a total takeover of the protocol's functionality, or other catastrophic failures. Critical vulnerabilities are non-negotiable; they absolutely must be fixed.

High (H): Significant Risk of Major Loss or Compromise

High-severity issues represent serious weaknesses that could result in significant financial losses for users, major malfunctions within the protocol, or substantial compromise of its intended operations. While exploiting these vulnerabilities might require specific conditions to be met or a moderate level of technical skill, the potential damage is considerable. These findings are critical and should be addressed and resolved thoroughly before the contract is put into the Mainnet.

Medium (M): Potential for Moderate Harm Under Specific Circumstances

Medium-severity bugs are loopholes in the protocol that could lead to moderate financial losses or partial disruptions of the protocol's intended behavior. However, exploiting these vulnerabilities typically requires more specific and less common conditions to occur, and the overall impact is generally lower compared to high or critical issues. While not as immediately threatening, it's still highly recommended to address these findings to enhance the contract's robustness and prevent potential problems down the line.

Low (L): Minor Imperfections with Limited Repercussions

Low-severity issues are essentially minor imperfections in the smart contract that have a limited impact on user funds or the core functionality of the protocol. Exploiting these would usually require very specific and unlikely scenarios and would yield minimal gain for an attacker. While these findings don't pose an immediate threat, addressing them when feasible can contribute to a more polished and well-maintained codebase.

Informational (I): Opportunities for Improvement, Not Immediate Risks

Informational findings aren't security vulnerabilities in the traditional sense. Instead, they highlight areas related to the clarity and efficiency of the code, gas optimization, the quality of documentation, or adherence to best development practices. These findings don't represent any immediate risk to the security or functionality of the contract but offer valuable insights for improving its overall quality and maintainability. Addressing these is optional but often beneficial for long-term health and clarity.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Severity Matrix

Impact

Likelihood	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

Impact

- **High** - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- **Low** - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

Medium Severity Issues

Lack of slippage on _transfer can lead to users paying more tax than they expected

Resolved

Path
HedgexDao.sol

Function Name
`_transfer`

Description
The Transfer function lack slippage protection against dynamic fee changes. There can be situations where malicious validators can queue **`setBurnTax()`** or **`setSupplyFloor()`** transactions before user transfers execute in the mempool. This causes users to pay higher burn taxes than expected when their transaction processes

Impact
Medium: Users experience unexpected burn tax costs on AMM swaps example(they expected to pay 0.1% fee but now have to pay 1%)
Loss of user funds due to higher-than-anticipated fees

Likelihood
High : Owner can change burn tax parameters instantly without timelock and validators can easily grief users

Recommendation
Add slippage protection parameters

Rounding leads to not paying burn taxes

Resolved

Path

HedgexDao.sol

Function Name

`_transfer`

Description

The burn tax calculation uses integer division which rounds down to zero for small transfer amounts. Users can exploit this by making multiple small transfers to avoid paying burn taxes entirely(this will happen on smaller transfers naturally also). The fee calculation (amount * burnTax) / FEE_DENOMINATOR will result in zero fees when the numerator is smaller than the denominator

Numeric Example

If burnTax = 10 (0.1%), transferring 999 tokens: (999 * 10) / 10000 = 0 → No fee
If burnTax = 50 (0.5%), transferring 199 tokens: (199 * 50) / 10000 = 0 → No fee
If burnTax = 100 (1%), transferring 99 tokens: (99 * 100) / 10000 = 0 → No fee

Impact: Medium

Users can bypass burn tax mechanism completely through dust transactions
Protocol loses intended fee revenue from legitimate transactions
Large holders can split transactions to minimize or eliminate fees
If token price appreciates significantly in the future, the missed burn fees represent substantial lost value

Likelihood

High: Simple to exploit - users just need to calculate transaction sizes that result in zero fees

Recommendation

Implement minimum fee amounts

Functional Tests

Some of the tests performed are mentioned below:

- Transfer function behavior with various burn tax rates
- Fee calculation accuracy across different transaction amounts
- AMM pair interaction and tax application logic
- Fee exclusion mechanism validation
- Owner privilege function testing

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Threat Model

Contract	Function	Threats
HedgexDao	_transfer	Fee bypass via rounding, Logic inconsistencies
HedgexDao	_transfer	Lack of slippage leading to extra fee pay

Closing Summary

This security audit examined the HedgeX DAO ERC20 token contract, focusing on the burn tax mechanism and transfer functionality.

The audit identified two distinct vulnerabilities related to fee calculation and protection mechanisms of medium severity. HedgeX DAO team resolved all the mentioned issues

Disclaimer

At QuillAudits, we have spent years helping projects strengthen their smart contract security. However, security is not a one-time event—threats evolve, and so do attack vectors. Our audit provides a security assessment based on the best industry practices at the time of review, identifying known vulnerabilities in the received smart contract source code.

This report does not serve as a security guarantee, investment advice, or an endorsement of any platform. It reflects our findings based on the provided code at the time of analysis and may no longer be relevant after any modifications. The presence of an audit does not imply that the contract is free of vulnerabilities or fully secure.

While we have conducted a thorough review, security is an ongoing process. We strongly recommend multiple independent audits, continuous monitoring, and a public bug bounty program to enhance resilience against emerging threats.

Stay proactive. Stay secure.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With seven years of expertise, we've secured over 1400 projects globally, averting over $3 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

7+ Years of Expertise	**1M+** Lines of Code Audited
50+ Chains Supported	**1400+** Projects Secured

Follow Our Journey

AUDIT REPORT

September 2025

For

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com