QuillAudits

AUDIT REPORT

January 2026

For

Table of Content

Executive Summary

Project Name	Holeinone
Protocol Type	ERC20 Token
Project URL	https://h1token.com/
Overview	contract implements a standard ERC-20 token with an integrated anti-bot protection mechanism. It leverages OpenZeppelin libraries for ownership, context handling, and safe arithmetic operations. The token supports all core ERC-20 functionalities, including transfers, approvals, and allowances. An external PinkAntiBot contract is invoked before transfers to mitigate bot and malicious activity. The anti-bot feature can be enabled or disabled by the contract owner as needed.
Audit Scope	The scope of this Audit was to analyze the Holeinone token Smart Contracts for quality, security, and correctness.
Source Code link	https://polygonscan.com/address/0x5494140d3CeeA77F75Df215E0f6C5238Ff52C812#code
Contracts in Scope	AntiBotStandardToken
Language	Solidity
Blockchain	Polygon
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	5th January 2026
Updated Code Received	8th January 2026
Review 2	8th January 2026

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	0(0.0%)
Low	0(0.0%)
Informational	1 (100%)

Severity

Issues	Critical	High	Medium	Low	Informational
Open	0	0	0	0	0
Acknowledged	0	0	0	0	**1**
Partially Resolved	0	0	0	0	0
Resolved	0	0	0	0	0

Summary of Issues

Issue No.	Issue Title	Severity	Status
1	External Anti-Bot Logic Integrated into ERC20 Transfer flow	Informational	Acknowledged

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.

Informational Issues

External Anti-Bot Logic Integrated into ERC20 Transfer flow

Acknowledged

Path

Holeinone.sol AntiBotStandardToken contract

Function

`_transfer()`

Description

The _transfer implementation includes conditional logic that invokes an external anti-bot contract when enableAntiBot is enabled. This anti-bot mechanism is integrated directly into the core transfer flow and is executed before balance updates occur.

The pinkAntiBot contract and its internal logic are explicitly out of scope for this audit and were not reviewed.

For the purpose of this audit, it is assumed that the external anti-bot contract is non-malicious, correctly implemented, and appropriately governed.

Impact

Although out of scope, integrating external logic into the transfer path introduces risk. Reverts or restrictive behavior in the anti-bot contract may block legitimate transfers, break composability with external protocols, or alter transfer behavior post-deployment due to governance, upgradeability, or misconfiguration.

Recommendation

Document the behavior of the anti-bot mechanism. If feasible, the external anti-bot contract should undergo a separate security review.

Functional Tests

Some of the tests performed are mentioned below:

- ✔ Should initialize correctly
- ✔ Should correctly mint to the deployer
- ✔ Transfers should reflect on the holder's and receiver's balance
- ✔ Should correctly grant approval to approved 3rd party

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

System Components & Trust Assumptions

Component	Description	Trust Level	Notes
AntiBotStandardToken	ERC20 token with Anti-Bot hook	Medium	Core logic is standard
Contract Owner	Controls anti-bot toggle	Medium–High	Centralized authority
PinkAntiBot	External transfer validator	Low	Fully trusted by token
Users (EOA)	Token holders	Untrusted	No special privileges
External Contracts	DEXs, bots, routers	Untrusted	Can interact via ERC20

Threat Enumeration

Threat	Description	Impact
External Call Reentrancy	AntiBot hook executes before state updates	State corruption / DoS
Transfer Censorship	AntiBot can revert any transfer	Frozen token
Centralized Control	Owner can toggle AntiBot anytime	Governance abuse
Malicious AntiBot	Untrusted AntiBot at deployment	Permanent lock
ERC20 Allowance Race	approve overwrite front-running	Token drain
ETH Transfer on Deploy	Fixed ETH transfer	Deployment revert
No Supply Cap	No explicit max supply	Future risk
Silent Transfer Failures	No AntiBot events	Poor UX

Attack Scenarios

Scenario	Description	Result
Malicious AntiBot	AntiBot blocks all sells	Token unusable
Reentrancy via AntiBot	Recursive token calls	Unexpected behavior
Selective Blacklisting	Targeted wallet blocks	Funds frozen
Owner Abuse	Toggle AntiBot post-launch	Trust loss

Closing Summary

In this report, we have considered the security of HoleinoneToken. We performed our audit according to the procedure described above.

No critical issues in Holeinone token, just 1 issue of Informational severity was found. The Holeinone Team acknowledged the issue.

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.

QuillAudits

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

Follow Our Journey

AUDIT REPORT

January 2026

For

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com